Development of an Optimization Framework for Solving Engineering Design Problems

A thesis submitted to the School of Mathematics at the University of East Anglia in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Chengyang Liu

September 2020

Abstract

The integration of optimization methodologies with computational simulations plays a profound role in the product design. Such integration, however, faces multiple challenges arising from computation-intensive simulations, unknown function properties (i.e., black-box functions), complex constraints, and high-dimensionality of problems. To address these challenges, metamodel-based methods which apply metamodels as a cheaper alternative to costly analysis tools prove to be a practical way in design optimization and have gained continuous development. In this thesis, an intrinsically linear function (ILF) assisted and trust region based optimization method (IATRO) is proposed first for solving low-dimensional constrained black-box problems. Then, the economical sampling strategy (ESS), modified trust region strategy and self-adaptive normalization strategy (SANS) are developed to enhance the overall optimization capability. Moreover, as the radial basis function (RBF) interpolation is found to better approximate both objective and constraint functions than ILF, a RBF-assisted optimization framework is established by the combination of the balanced trust region strategy (BTRS), global intelligence selection strategy (GIS) and early termination strategy (ETS). Following that, the fast computation strategy (FCS) and successive refinement strategy (SRS) are proposed for solving large-scale constrained black-box problems and the final optimization framework is called as RATRLO (radial basis function assisted and trust region based large-scale optimization framework). By testing a set of well-known benchmark problems including 22 G-problems, 4 engineering design problems and 1 high-dimensional automotive problem, RATRLO shows remarkable advantages in achieving high-quality results with very few function evaluations and slight parameter tuning. Compared with various state-of-the-art algorithms, RATRLO can be considered one of the best global optimizers for solving constrained optimization problems. Further more, RATRLO provides a valuable insight into the development of algorithms for efficient large-scale optimization.

Acknowledgements

First and foremost, I would like to thank my parents and sister for all their love, support and sacrifices. I am extremely grateful for what you have done not just during my PhD, but over the past twenty six years. It is the most luckiest thing for me to be born in this family. I am always happy and carefree because you undertake all the hardships of life silently all the time. Without your encouragement and support, I would never have completed this long journey and pushed my self to highest of my abilities.

Next, I would express my sincere gratitude to Dr Dianzi Liu, my supervisor, for his valuable guidance and mentorship throughout my study. Thank you for your patience and for making time to carefully work through problems with me whenever I was stuck or confused, even with the most simple things. Thank you also for revising and improving my academic papers without any complaining and always encouraging me to do better. It is your encouragement and trust that makes me get through all the difficulties in the study. This is the best thing forever that I could always pour out my bitterness and trouble to you and be honest to say 'I have done nothing in the last few weeks because ...'. I am lucky to be your student and am proud to realize my potentials following your excellent supervision.

I would also like to thank Dr Xiao'an Mao for all the advice and support. Thank you for the timely and helpful feedback during my study. And thank you for the conference recommendations and for helping me prepare the manuscript. I am sorry that I did not carry out further research in thermoacoustic systems but I am very grateful for your sincere instructions.

A special acknowledgement goes to Dr Dane Grundy and Ms Rebecca Hill. Thank you for inviting and coaching me to play squash despite my bad oral English. It helps me de-stress and makes my PhD experience much better.

Last but by no means least, I would like to thank Mr Chenghao Yue, Mr

Weijiao Zhang, Ms Junyu Liu and Ms Nan Cheng. In most of the weekends, we shared the time together. One of the happiest thing during my PhD is to play cards with you. Thank you for constantly making me laugh and keeping me sane during four years.

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

Nomenclature

Symbols

$[\delta_g, \delta_f]$	The normalization coefficients used in SANS (self-adaptive normalization strategy)
$\boldsymbol{X}$	The set of fitting points
$\boldsymbol{x}$	The design vector
Δ^0	The relative size of the initial trust region
$\Delta_{min,2}$	The tolerated relative size of the trust region used in ETS (early termination strategy)
Δ_{min}	The minimum relative size of the trust region
Δext	The relative size of the extended box
η_{eco}	The economical factor used in ESS (economical sampling strategy)
η_{SRS}	SRS (successive refinement strategy) will not activate in the first $\eta_{SRS} \cdot NFEs_{max}$ function evaluations
$\lfloor \cdot \rfloor$	Round down to the nearest integer
d	The number of design variables
$f(\boldsymbol{x})$	The objective function
$g(\boldsymbol{x})$	The constraint function
$I_{max,2}$	The tolerated variation in objective function values used in SRS (successive refinement strategy)
I_{max}	The tolerated variation in objective function values used in ETS (early termination strategy)

K_{max}	The maximum number of iterations
k_{RES}	Used in BTRS (balanced trust region strategy), the subspace of a design variable will only be shrunk in the first k_{RES} iterations when it is located on the global bounds
m	The number of constraint functions
n_{ext}	The number of points located in the extended box
N_{plan}	The default number of required sampling points
$NFEs_{max}$	The maximum number of function evaluations
Q	The design space
$SRS_{feasible}$	Control the termination of SRS (successive refinement strategy) when the iteration point is feasible
t_{max}	The maximum number of iterations in SRS (successive refinement strategy)
TOL_{con}	The constraint tolerance

Abbreviations

ANFEs	The average of the number of function evaluations
AREs	The average of the number of function evaluations required to find the best solution
BTRS	Balanced trust region strategy
DOE	Design of experiments
EAREs	The efficient value of AREs
ENFEs	The efficient number of function evaluations
ESS	Economical sampling strategy
ETS	Early termination strategy
FCS	Fast computation strategy
FR	Feasible rate
GIS	Global intelligence selection strategy
IATRO	Intrinsically linear function assisted and trust region based optimization method

IEMS	IATRO–ESS–MTRS–SANS
ILF(R)	Intrinsically linear function (regression)
MAM	Multipoint approximation method
MBDO	Metamodel-based design optimization
MTRS	Modified trust region strategy
PVD	Pressure vessel design
RATR(L)O	Radial basis function assisted and trust region based (large) optimization framework
RBF(I)	Radial basis function (interpolation)
RESBGEFS	RBFI–ESS–MTRS–SANS–BTRS–GIS–ETS–FCS–SRS
SANS	Self-adaptive normalization strategy
SQP	Sequential quadratic programming
SR	Success rate
SRD	Speed reducer design
SRS	Successive refinement strategy
TE	Termination efficiency
TSD	Tension/compression spring design
WBD	Welded beam design

1

Introduction

1.1 Background

People optimize. Investors aim to find an optimal product that avoids excessive risk while achieving highest returns. Doctors are committed to improving medical skills in order to ease the body and financial burden of patients. Engineers adjust parameters for the purpose of perfecting the performance of their designs.

Nature optimizes. The micro and nanoscopic architectures on the surface of lotus leaves minimizes the droplet's adhesion to that surface, which makes lotus clean and elegant. Bees build honeycombs with hexagon cells that have maximum volume with minimum cost to support reproduction. Closed physical systems tend to reach equilibrium states with the minimum internal energy.

In manufacturing industries, optimization is such an important tool for addressing the global competitions to produce better and cheaper products. As shown in Fig 1.1, a typical process of design optimization includes three stages.

The first stage is the modelling stage, in which the variables, objectives and constraints are determined. The variables are the certain characteristics of the design such as the temperature, the length, or the volume of the design. In some way, the variables are often restricted or constrained. The objective or the constraint is a quantitative measure of the performance of the design under study. It could be the cost, the fluid behavior, the mechanical property or any quantity or combination of quantities that can be represented by a single number. Usually in engineering, the objective or the constraint value is related to the complex physical phenomena of the system and is evaluated by numerical simulations such as finite element method

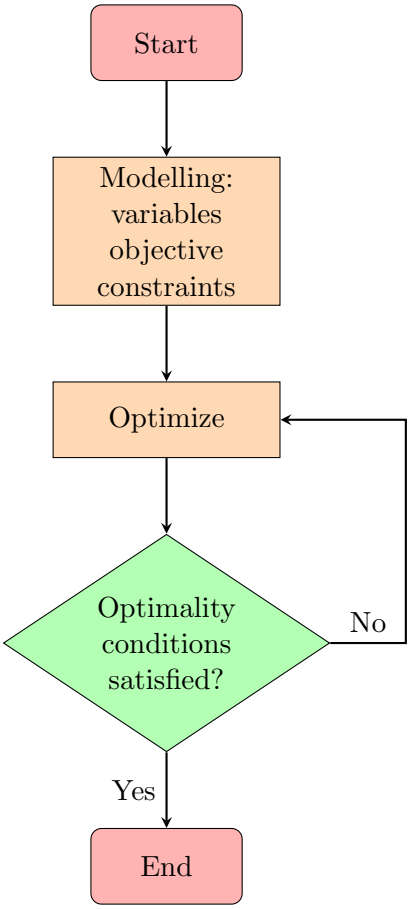


Figure 1.1: Schematic of design optimization in engineering

(FEM) or finite difference method (FDM). Generally, both the objective and the constraints are dependent on the variables and the goal is to find a set of variables that optimize the objective while satisfying the constraints.

Once the model has been formulated, an optimization algorithm can be applied to find the solution. As well documented in 'no free lunch' theorem [1], there is no universal optimization algorithm but rather an algorithm that is tailored to a particular type of optimization problem. It is important to choose an appropriate optimization algorithm as it determines directly if the solution of the problem can be found at all.

After a solution has been found by the algorithm, the designer must recognize whether the solution is indeed the optimum of the problem by checking some optimality conditions [2]. And if the optimality conditions are not satisfied, they may provide useful information on how the current solution can be improved and suggest ways in which the model can be refined or corrected. As a result, the optimization problem is solved again and the aforementioned process repeats.

In general, there are three major challenges in the real-world design optimization process. First, the numerical simulation in the modelling stage is usually a 'black box', i.e., the explicit mathematical expressions of the objective and constraint functions are unavailable, along with their corresponding derivatives. Hence, gradient-based optimization algorithms such as the sequential quadratic programming (SQP) and interior-point method [2] are hardly to be used. Instead, derivative-free algorithms [3] and metaheuristic algorithms (MA) such as genetic algorithms (GA) [4], particle swarm optimization (PSO) [5] and ant colony optimization [6] are developed for optimizing this kind of black-box problems. Nevertheless, the enormous computational overhead of complex high-fidelity engineering simulations [7] makes it impractical to depend exclusively on simulations for design optimization. In [8], it is reported that it takes Ford Motor Company about 36-160 hrs to run one crash simulation. Therefore, a good optimization algorithm should solve expensive optimization problems within a severely limited number of evaluations. Last but not the least, another challenge in addressing expensive black-box optimization problems, is the limitations imposed by the existing constraints, restricting the valid solutions to a smaller subset of the design space.

Hence, it is meaningful and valuable for scientists to develop an efficient optimization framework for solving expensive constrained black-box optimization problems.

1.2 Related work

In the past three decades, in order to apply optimization algorithms in solving problems involving black-box simulations, metamodel (also called surrogate model) methods have aroused broad attention. Generally, a metamodel is an approximation of a detailed simulation model, i.e., a model of a model [9]. Mathematically speaking, a metamodels is an analytical description created based on a dataset of input and the corresponding output from a detailed simulation model. The mathematical description could vary depending on the intended use or the underlying physics that the model should capture. With a metamodel, various optimization methods can then be applied to search for the optimum, which is therefore referred as metamodel-based design optimization (MBDO) or surrogate-assisted design optimization (SADO) [9, 10].

From in-depth review on MBDO by Barthelemy et al. [11], Haftka et al. [12] and Wang et al. [13], the benefits of MBDO can be summarized as follows:

1. It is easier to connect the expensive and proprietary simulation codes.
2. Parallel computation is simpler to be implemented as it involves running the same simulation code at various design points.
3. Usually, numerical noise existed in simulation can be well filtered by constructing specific metamodels.
4. The metamodel can reveal the hidden properties of the black-box simulation and helps researchers to determine the promising design space where new high-fidelity experiments should be run.
5. If a global metamodel can be built, it renders a view of the entire design space which makes it easier to analyze the simulation process.

Because of the practicability of MBDO, new developments have been continuously coming forth in the literature. On the whole, these developments can be classified into three major categories.

One category is the advancement in design of experiments (DOE). The theories of DOE originate from planning physical experiments. Classic experimental designs focus on controlling the random error in physical experiments so that the results can have minimum influence in the approval or disapproval of a hypothesis. Therefore, the classic DOE tends to scatter the sampling points around boundaries of the search space and leaves a few at the center of the search space. Widely used classic kinds of DOE comprise factorial or fractional factorial, central composite, Box-Behnken, alphabetical optimal, and Plackett-Burman designs, see Myers et al. [14]. However, computer experiments involve mainly the systematic error rather than the random error in physical experiments. There seems to be a consensus among scientists that a proper experimental design for deterministic computer analyses should be space-filling, which aims to fill the complete design space rather than to concentrate on the boundary [13, 15, 16]. Popular space-filling designs include Latin hypercube designs (LHS) [17], minmax and maxmin designs [18], Hammersley sequence sampling [19], uniform designs [20] and orthogonal arrays [21]. See [22] for more details about these designs. In [23], the authors stated that the Latin hypercube design is only uniform in 1-D projection while the other methods tend to be more uniform in the entire space. In general, the accuracy of a metamodel can be improved by increasing the number of sampling points. But for low-order polynomial metamodels this is only valid up to a certain limit. Thereafter, increasing the number of points is not beneficial to the approximation accuracy.

Another category is the progress in metamodel building. Up to now, commonly used and studied metamodels include polynomial regression (PR) [24], radial basis function (RBF) [25], Kriging [26], multivariate adaptive regression splines (MARS) [27], artificial neural networks (ANN) [28] and support vector regression (SVR) [29]. Comparative studies have been made over the past years. For example, Jin et al. [15] investigated and compared PR, Kriging, MARS and RBF models. The authors claimed that the RBF metamodel outperforms others in most instances, especially in shortage of computational resource (the sample size is small). As the growth of fitting sets, the performance of Kriging and MARS models will gain improvement. In addition, Kriging is sensitive to the noise but PR performs well in this situation. In a comparison with PR, Kriging, MARS and RBF metamodels, Clarke et al. [29] found that SVR had the superior performance in regard to accuracy and robustness with the manually optimized Gaussian kernel function. In contrast, Kim et al. [30] compared moving least squares method (MLS), Kriging, RBF and SVR metamodels and concluded that Kriging and MLS build more accurate metamodels than RBF and SVR models. Therefore, it is unreasonable to draw any decisive conclusions on the superiority of any of the mentioned metamodels. The quality of a metamodel can vary considerably depending on how many design variables are involved, what types of the fitting functions are, and how well the predefined parameters are tuned.

Last but not the least, various searching schemes have been developed with support of metamodels.

One popular approach in MBDO is to embed the metamodels in the framework of metaheuristic algorithms (MA). Generally, Kriging models and RBF models are widely used to accelerate the convergence of MA in two ways. First, metamodels are applied to prescreen promising candidate offspring points produced by the basic MA. For Kriging-based MA, the prescreen criterion can be based on the expected improvement [31, 32], the probability of improvement [33], the lower confidence bound [34] and the multi-objective infill criterion [35]. For RBF-assisted MA, the predicted response of surrogate is used for prescreening. This can be found in RBF-assisted PSO algorithm [36–39], RBF-assisted GA [40], and RBF-assisted differential evolution [41, 42]. Second, the predicted optimum provided by the surrogate model is used to replace the current candidate offspring point. Take the PSO framework as an example, the global best point may be replaced by the optima obtained by the global RBF metamodel [43–45]. Similar applications can be found in [32, 46–51]. Moreover, the

personal historical best particle in PSO can be refined through the local surrogate model within the vicinity of this point, which is applied in [52–54]. Similar approaches can be found in other surrogate-assisted MA [43, 55–59].

Another approach is based on the uncertainty of the surrogate model to direct the optimization process to explore the complete design space. This approach is mainly used in Kriging-based optimization algorithms, for example the efficient global optimization (EGO) proposed by John et al. [60]. Kriging not only provides an estimate of the original function everywhere, but also a normal distribution around that value that characterizes the uncertainty [61]. In EGO, a global Kriging metamodel is built at first, and then, additional points are sampled that maximize the expected improvement (EI) over the present best solution. These 'exploration' points improve the accuracy of the approximation around the global optimum, which is quite effective in solving low-dimensional problems [62]. Krause et al. [63] proposed another uncertainty-based criterion, which is to maximize the surrogate prediction minus a multiple of the prediction variance. Moreover, another way of utilizing the surrogate prediction and its prediction variance is to maximize the expected posterior information gain about the global maximizer, which is referred as entropy based search strategies [64–66].

Without using an uncertainty structure, there are methods that use other approaches to balance exploitation and exploration. Regis et al. [67] proposed a distance-based searching strategy, where the optimum of the surrogate is obtained under the condition that it is at a given minimum distance from any of the previous simulation points. The value of this distance parameter controls the behavior of the optimization process, i.e., whether the process is exploring (large value) or exploiting (small value) the design space. Wang et al. [68] developed the mode-pursuing sampling approach (MPS), which samples points preferentially where the surrogate has low values using a probability function. A region elimination strategy was presented by Younis et al. [69], which explores the design space by removing the less promising and previously searched regions. The promising regions are identified by dividing the design space into many subspaces and carrying out search in each one of these subspaces. Similar schemes can be found in [70–74]. Another popular searching scheme is the trust region or move limit strategy where the surrogate model is constructed in successive regions of interest, centering on the optimum obtained in the last iteration. During the optimization process, the quality of the approximation increases as the trust region becomes much smaller and meaningful. This strategy was applied in

the framework with the moving least squares regression metamodel [75, 76] and the intrinsically linear function regression metamodel [77–80].

Although the above-mentioned algorithms can obtain good results on black-box problems with boundary constraints, most of them have difficulty dealing with nonlinear constrained optimization problems. As stated by Haftka et al. [61] and Muller et al. [81], the state of the art about constrained optimization is less advanced. The main challenge lies in the definition of a general convergent scheme that seeks a feasible and optimized solution under a reasonable number of function evaluations. The hurdles become even more distinct as the number of constraints increases or when discontinuous responses are presented [82]. Generally, there are four widely-used approaches for constraint handling [83–89]:

(i) unconstrained optimization using a penalty function (penalizing the fitness value of infeasible solutions),
(ii) feasible solution preference methods,
(iii) Repair algorithms to generate feasible solutions from infeasible ones,
(iv) multi-objective optimization, where the constraint functions are defined as additional objectives.

The penalty-based approach (i) which lumps all constraint functions into one penalty function is the most frequently used approach due to its easy-implementation [44, 90, 91]. But the main drawback is that the information of the individual constraint is lost and additional parameters need to be fitted for penalty terms [92]. It has difficulty solving a problem in which the optimum lies in the boundary between feasible and infeasible regions or when the feasible region is disjoint [93]. In feasible solution preference methods (ii) [94–97], too little information from infeasible solutions is involved in the optimization process, which increases the risk of the optimization process getting stuck in local optima. And by investigating several repairing algorithms (iii) in [98–101], it can be concluded that defining a general scheme to reduce the constraint violation can be as complex as solving the problem itself. Recent studies about treating the constraints as one or more objective functions to be optimized (iv) can be found in [31, 87, 102, 103]. Since additional parameters might be included to weigh these objectives, the optimization performance of these methods is more or less unstable.

Besides, relatively few algorithms can handle expensive constrained black-box optimization problems under severely limited budget. Take the well-known G-problem or G-function benchmark, which was introduced by

Michalewicz [83] and Floudas [104] as an example, the number of function evaluations (NFEs) required for lots of algorithms is not realistic. Probably hundreds of thousands of function evaluations were required in algorithms such as SAFF [105], COPSO [106], ISR [107], ATIMES [108], SMES [97], ECHT-EP2 [109], HCOEA [110], αSimplex [111], HCS–LSAL [112], LCA [113] and more. Some algorithms manage to use fewer function evaluations to solve G-problems. For example, Zahara and Kao [100] tested G04, G08, G12 and they can be solved within 20000 evaluations. Regis [114] proposed the ConstrLMSRS that uses RBF surrogate to model objective and constraint functions separately. ConstrLMSRS was able to quickly obtain feasible solutions of 9 problems (G02 to G10) but a feasible starting point should be given to start the optimization process. In addition, Regis [115] developed COBRA, an efficient solver which still makes use of RBF interpolation to approximate objective and constraint functions but the new iterate is selected according to the violation of constraints within some small margin. Good results were achieved on 13 G-problems (G01, G03, G05, G06, G07, G08, G09, G10, G13, G16, G18, G19, G24) when the starting points were infeasible. But the objective and constraint functions of three problems (G03, G05, G13) had to be manually rescaled to avoid difficulty in fitting RBF surrogates. For the purpose of enhancing the performance of COBRA, Bagheri et al. [116] presented a SACOBRA – self-adjusted version of COBRA. SACOBRA is very efficient that 8 G-problems (G01, G03, G04, G05, G06, G07, G08, G11) can be solved within 500 function evaluations. But the obtained solutions were not optimal enough. Jiao et al. [87] introduced a self-adaptive selection strategy into the evolutionary algorithm which combines feasibility with multi-objective problem techniques. 22 G-problems were tested and some of them (G05, G06, G08, G11, G12, G18) can be solved very efficiently in less than 1000 evaluations, but others require 5000-100000 evaluations to be solved. Dong [74] proposed a Kriging-based constrained global optimization algorithm SCGOSR with space reduction strategy. In SCGOSR, new added samples are selected from optimal solutions obtained by the multi-start solver. Five problems (G04, G06, G07, G08 and G09) can be well optimized by SCGOSR within hundreds of function evaluations but the obtained solutions were just around the global optimum. Similarly in KCGO developed by Li et al. [117], near-optimal solutions of seven problems (G04, G06, G07, G08, G09, G10 and G12) can be found within 200 function evaluations. Liu et al. [118] proposed the eDIRECT-C algorithm which employs an adaptive metamodeling strategy to build appropriate metamodel types for objective and constraints respectively. Thirteen G-problems (G01 to G13) were tested and nine of them (except G02, G05, G09 and G13) can be successfully solved within 1000 function

evaluations. Besides, the design space of G08 and G12 have to be manually reset because of the shortcomings in the sampling strategy.

Moreover, few strategies have been developed to aim for solving expensive, high-dimensional (more than a hundred decision variables) and severely constrained problems. There seems to be a critical lack of research on large-scale problems in MBDO, and many questions are not answered or even addressed [13]. A well-known case of the large-scale problem is the MOPTA08 automotive application with 124 decision variables and 68 black-box inequality constraints [119]. The pioneer work in [119] revealed the huge difficulty in solving this problem by testing various algorithms on this problem. The optimization progress was either extremely slow or fast enough but no feasible solutions can be found. Recently, good results have been obtained by ConstrLMSRS [114], COBRA [115] and SACOBRA [116]. And among them, the SACOBRA algorithm can be viewed as the state of the art in solving this problem but it is not so competitive in solving low-dimensional problems.

Therefore, it is meaningful and valuable to develop an efficient optimization framework for solving expensive constrained black-box optimization problems. Specifically, the objectives of the research consist of three aspects. First, it is critical to develop a balanced search scheme which could not only explore the global design space but also exploits the specific design space for the global optimum of the problem. Second, it is meaningful to determine which type of the metamodel should be used during the search to replace the costly black-box model. Finally, it is highly desired to make full use of each function evaluation for the purpose of reducing the computation cost. Also note that these aspects are not independent and should be considered together in the development of the optimization framework.

1.3 Thesis outline

In this thesis, the important concepts and fundamentals of metamodel-based optimization are explained and discussed in Chapter 2. In Chapter 3, the development process of IATRO (intrinsically linear function assisted and trust region based optimization method) is demonstrated in details. 26 benchmark problems are tested and the results are compared with the solutions obtained by various metaheuristic algorithms and metamodel-based algorithms. Based on IATRO, several new strategies including the economical sampling strategy (ESS), the self-adaptive normalization strategy

(SANS) and the modified trust region strategy (MTRS) are developed to enhance the overall optimization performance. The enhanced IATRO is called EIATRO and the details are described in Chapter 4. To further improve the performance of EIATRO, the radial basis function (RBF) interpolation is employed to build the metamodels instead of the intrinsically linear approximation. Chapter 5 presents this new optimization framework entitled RATRO (radial basis function assisted and trust region based optimization framework) with integration of the balanced trust region strategy (BTRS), the global intelligence selection (GIS) strategy and the early termination strategy (ETS). After this chapter, a large-scale optimization framework RATRLO (radial basis function assisted and trust region based large-scale optimization framework) is proposed in Chapter 6. RATRLO aims at solving large-scale optimization problems but also works excellently in optimizing low-dimensional problems. The main developments in RATRLO include the fast computation strategy (FCS) and the successive refinement strategy (SRS). And the final optimization results on one large-scale optimization problem (the MOPTA08 problem [119]) and 26 low-dimensional problems are also discussed. Finally, the thesis is ended with discussions regarding the proposed optimization framework, conclusions and an outlook of further needs.

2

Basics

2.1 Mathematical formulation of constrained black-box optimization (CBO) problems

Mathematically speaking, optimization is the minimization or maximization of a function subject to constraints on its variables. In this thesis, we use the following notation:

- $\boldsymbol{x}$ is the vector of d variables $(x_1, x_2, ..., x_d)$;
- $f(\boldsymbol{x})$ is the objective function, a (scalar) function of $\boldsymbol{x}$ that we want to maximize or minimize;
- $g_j(\boldsymbol{x})(j = 1, ..., m)$ are m constraint functions, which are (scalar) functions that the unknown vector $\boldsymbol{x}$ must satisfy.
- Q is the design space in the range of $[\boldsymbol{A}, \boldsymbol{B}]$, where $A_i\,(i = 1, ..., d)$ and $B_i\,(i = 1, ..., d)$ are the given lower and upper bounds on the design variable x_i;

Using this notation, the constrained black-box optimization (CBO) problem can be written as

$$\begin{aligned} &\min_{\boldsymbol{x} \in Q} f(\boldsymbol{x}) \\ &\text{s.t. } g_j(\boldsymbol{x}) \leq 0 \ (j = 1, ..., m) \\ &\qquad A_i \leq x_i \leq B_i \ (i = 1, ..., d) \end{aligned} \tag{2.1.1}$$

Here, it is noteworthy that there are several assumptions for this formulation which can be described as follows:

1. For simplicity, the equality constraint function $h(\boldsymbol{x}) = 0$ can be

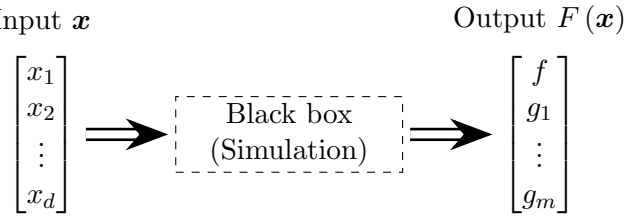


Figure 2.1: Black-box simulation

converted into the inequality form $|h(\boldsymbol{x})| \leq \epsilon$ where ϵ is a small positive constraint tolerance (the default value is $1e-6$). In other words, one equality constraint function will be replaced by two inequality constraints: $g(\boldsymbol{x}) = h(\boldsymbol{x}) - \epsilon \leq 0$ and $g(\boldsymbol{x}) = -h(\boldsymbol{x}) - \epsilon \leq 0$.

2. $f(\boldsymbol{x})$ and $g_j(\boldsymbol{x})\,(j = 1, ..., m)$ are deterministic black-box functions where there is no noise involved. This is to say, there are no algebraic expressions of both the objective and the constraint functions. A set of design variables $x \in Q \subseteq \mathbb{R}^{\mathrm{d}}$ are inputted to the black-box, e.g., a simulation tool, and a certain set of responses $F(x) \subseteq \mathbb{R}^{\mathrm{m+1}}$ are the output based on the unknown relationship between the variables and responses. This process is illustrated in Fig 2.1 and is assumed to be computationally expensive so that the derivative information is also unavailable or impractical to obtain.

3. Equation 2.1.1 describes the minimization problems because maximizing f can be easily transformed in the form of minimizing $-f$ without loss of generality.

4. Although the objective and constraint functions are black-box, we assume that the values $f(\boldsymbol{x})$ and $g_j(\boldsymbol{x})\,(j = 1, ..., m)$ for any input $\boldsymbol{x} \in [\boldsymbol{A}, \boldsymbol{B}]$ can be obtained without any crash on the simulator.

2.2 Design of experiments (DOE)

As mentioned earlier, a space-filling design which distributes points everywhere in the design space with as few gaps as possible is preferred in MBDO. Several popular designs of experiments are introduced in this section.

2.2.1 Minmax and maxmin distance designs

Minmax distance design

Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ be the n experimental design points, where each point $\boldsymbol{x}_i \in Q$. Then, for any point $\boldsymbol{x} \in Q$, we can find the distance to the nearest design point as $\min_i \boldsymbol{d}(\boldsymbol{x}, \boldsymbol{x}_i)$, where $\boldsymbol{d}(\boldsymbol{u}, \boldsymbol{v}) = (\sum_{i=1}^{n} |u_i - v_i|_s)^{1/s}$. Here $s = 1$ and $s = 2$ means the rectangular and Euclidean distances respectively. In the entire design space, if we define the worst point is the point farthest from all the design points, the distance of the worst point to the nearest point in $\boldsymbol{X}$ can be obtained by $\max_{\boldsymbol{x} \in Q} \min_i \boldsymbol{d}(\boldsymbol{x}, \boldsymbol{x}_i)$. Therefore, a minmax space-filling design [18] can be obtained by minimizing this worst distance as

$$\min_{\boldsymbol{X}} \max_{\boldsymbol{x} \in Q} \min_i \boldsymbol{d}(\boldsymbol{x}, \boldsymbol{x}_i) \tag{2.2.1}$$

Maxmin distance design

Another instinctive option to create a space-filling design is to scatter the points in the entire experimental space so that they are as far apart as possible. This can be found by maximizing the minimum distance among the points in $\boldsymbol{X}$ as:

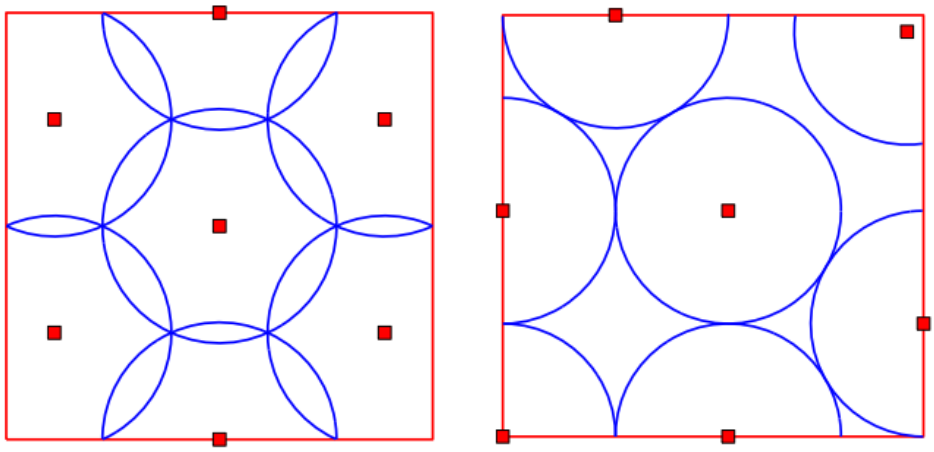
$$\max_{\boldsymbol{X}} \min_{i,j} \boldsymbol{d}(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.2.2}$$

Apparently, a maxmin distance design is much easier to implement than a minmax distance design because the minmax distance design involves computing the distances for the whole design space while for a maxmin distance design, we only have to take the distances among the design points into considerations.

A 7-point minmax and maxmin distance design for two variables using the Euclidean distance metric is shown in Fig 2.2. It can be seen clearly that for small d, a minmax distance design tends to distribute points in the interior of the design space while a maxmin distance design will generally result in the points lying on the exterior of the experimental region.

2.2.2 Latin hypercube design (LHD)

In statistical sampling, a square gird that contains sample positions is a Latin square if and only if there is only one sample in each row and each column. A



(a) minmax distance design (b) maxmin distance design

Figure 2.2: A 7-point minmax and maxmin distance designs in a square from [120]

Latin hypercube is the generalization of this concept to multiple dimensions. In each axis-aligned hyperplane, each sample is unique.

For a basic Latin hypercube design of d variables, the range of each variable $[A_i, B_i]$ is divided into n non-overlapping intervals of equal probability. One value from each interval is randomly selected but with regard to the probability distribution in the interval. Then n points can be constructed to satisfy the Latin hypercube requirements [121]. This generates an $n \times d$ sampling matrix $\boldsymbol{X}$, where the d columns are the levels of each variable, and the n rows describe the specific parameter settings for each design.

Mathematically, the first step to create a basic Latin hypercube design is to generate a matrix $\boldsymbol{P}$ with elements

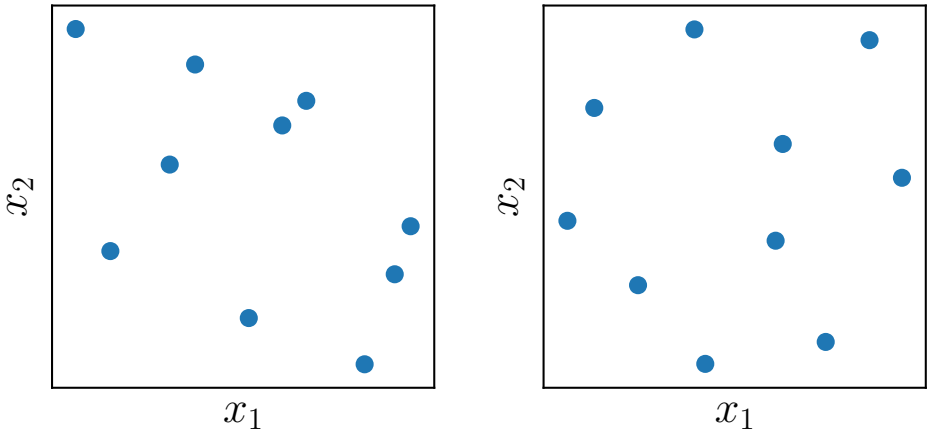
$$\boldsymbol{P}_{i,j} = \frac{\pi_j(i) - U_{i,j}}{n} \ (1 \leq i \leq n, 1 \leq j \leq d) \tag{2.2.3}$$

where $\pi_j(i), ..., \pi_j(n)$ are random permutations of the integers 1 to n and $U_{i,j}$ is an independent uniformly distributed random variable in the range $[0, 1]$. Then, the actual design matrix $\boldsymbol{X}$ can be obtained by

$$\boldsymbol{X}_{i,j} = F_{x_j}^{-1}(\boldsymbol{P}_{i,j}) \tag{2.2.4}$$

where $F_{x_j}^{-1}$ represent the inverse of the cumulative distribution function for variable j.

A 10-point basic Latin hypercube design for 2 variables is shown in Fig 2.3a Besides, there are several variants of Latin hypercube designs based on certain optimization criterion. The most popular approach to generate a better space-



(a) basic Latin hypercube design (b) maxmin Latin hypercube design

Figure 2.3: A 10-point basic and maxmin Latin hypercube designs for two variables

filling Latin hypercube design is to maximize the distances between any two points and this maxmin Latin hypercube design is shown in Fig 2.3b. For other Latin hypercube designs, please refer to [16].

2.2.3 Hammersley design

As described by Kalagnanam [122], Hammersley design belongs to a group called low-discrepancy sequences. The discrepancy is a measure of the difference from a uniform distribution and could be measured in several ways. In the following, we will give a brief mathematical formulation of Hammersley design. For more mathematical details, readers are referred to the literature [123].

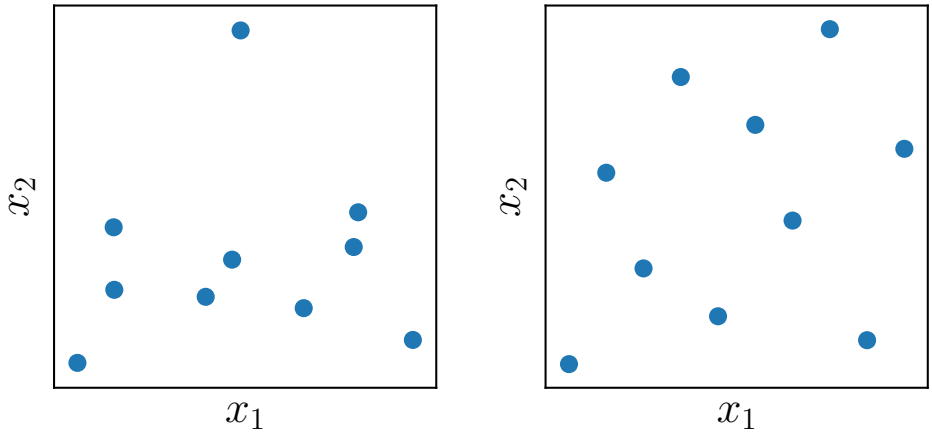
Each nonnegative integer l can be expressed by using a prime base p:

$$l = a_0 + a_1 p + a_2 p^2 + ... + a_r p^r \tag{2.2.5}$$

where each a_i is an integer in $[0, p-1]$. A function φ_p of l can be defined by

$$\varphi_p(l) = \frac{a_0}{p} + \frac{a_1}{p^2} + ... + \frac{a_r}{p^{r+1}} \tag{2.2.6}$$

For a design space of d variables, any sequence $p_1, p_2, ...p_{d-1}$ of prime numbers defines a sequence $\varphi_{p_1}, \varphi_{p_2}, ..., \varphi_{p_{d-1}}$ of functions. Then, the l-th d-dimensional Hammersley design is



(a) Monte-Carlo design (b) Hammersley design

Figure 2.4: A 10-point Monte-Carlo and Hammersley designs for two variables

$$\left(\frac{l}{n}, \varphi_{p_1}(l), \varphi_{p_2}(l), \varphi_{p_{d-1}}(l)\right) \quad l = 0, 1, 2, ..., n-1 \tag{2.2.7}$$

where $p_1 < p_2 < ... < p_{d-1}$ and n is the required number of sampling points. For specific formulations of function $\varphi_p(l)$, please refer to [124].

A typical Hammersley design for two variables with comparison to the classic Monte-Carlo design is shown in Fig 2.4.

2.3 Metamodels

As previously mentioned, a metamodel aims at approximating a detailed and usually computation-intensive simulation model, i.e., a metamodel is a 'model of a model'. In optimization, lots of evaluations on simulation model are required and each evaluation might be time-consuming, which makes a large number of optimization algorithms impractical to use. But metamodels can be used as surrogates for the detailed simulation model when appropriate, which decreases the computational cost a lot and enables the use of typical gradient-based algorithms in design optimization.

As described in Fig 2.1, a black-box simulation can be seen as a function $F : \mathbb{R}^d \mapsto \mathbb{R}^{m+1}$ which means that the function F maps the set of d design variables into another set of $m + 1$ responses:

$$\boldsymbol{y} = F(\boldsymbol{x}) \tag{2.3.1}$$

For each response in $\boldsymbol{y}$, a metamodel can be built to approximate this response

as

$$\tilde{y} = s(\boldsymbol{x}) \tag{2.3.2}$$

where $s(\boldsymbol{x})$ is the mathematical function defining the metamodel which approximates the observed response y as the predicted response $\tilde{y}$. In general, this approximation introduces the approximation error ϵ into y, i.e.

$$y = \tilde{y} + \epsilon = s(\boldsymbol{x}) + \epsilon \tag{2.3.3}$$

Generally, a metamodel is built from a dataset of input $\boldsymbol{x}_i$ and corresponding response values $y_i = f(\boldsymbol{x}_i)$, where $i = 1, 2, ..., n$ and n is the number of design points used to fit the model. Usually, these n fitting points have different variable settings $\boldsymbol{x}_i = (x_1, x_2, ..., x_d)^T$ of the d design variables.

In the following sections, several well-known metamodels are presented and their main characteristics as well as the basic theoretical ideas are discussed.

2.3.1 Polynomial metamodel

Polynomial metamodels, also referred to as response surface models (RSM) are developed using regression, which is the process of fitting a model $y = s(\boldsymbol{x}, \boldsymbol{\beta}) + \epsilon$ to a dataset of n variable designs $\boldsymbol{x}_i$ and corresponding responses y_i.

The regression coefficients $\boldsymbol{\beta}$ are determined by the least squares method, i.e., by solving the minimization regression problem

$$min \sum_{i=1}^{n} \epsilon_i^2 = \min_{\beta} \sum_{i=1}^{n} (y_i - s(\boldsymbol{x}_i, \boldsymbol{\beta}))^2 \tag{2.3.4}$$

The most popular polynomial metamodels are linear regression models where $s(\boldsymbol{x}, \boldsymbol{\beta})$ is linear in β and consists of low order polynomials. For instances, the

following models can be used to fit a metamodel in d design variables.

$$y = s(\boldsymbol{x}, \boldsymbol{\beta}) + \epsilon = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \epsilon \tag{2.3.5}$$

$$y = s(\boldsymbol{x}, \boldsymbol{\beta}) + \epsilon = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \sum_{i=1}^{d-1} \sum_{i<j=2}^{d} \beta_{ij} x_i x_j + \epsilon \tag{2.3.6}$$

$$y = s(\boldsymbol{x}, \boldsymbol{\beta}) + \epsilon = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \sum_{i=1}^{d-1} \sum_{i<j=2}^{d} \beta_{ij} x_i x_j + \sum_{i=1}^{d} \beta_{ii} x_i^2 + \epsilon \tag{2.3.7}$$

Equation 2.3.5, 2.3.6 and 2.3.7 are first-order, first-order with interaction, and second-order polynomial models respectively. Only second-order interaction effects, i.e., effects that involve just two variables are included in the presented models.

In matrix notation, the process of finding the least square estimates can be briefly described. The regression model can be written as

$$\boldsymbol{Y} = \boldsymbol{X} \cdot \boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.3.8}$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \vdots \end{bmatrix}, \boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{2.3.9}$$

$$\begin{aligned} \boldsymbol{P} &= \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} & x_{11}x_{12} & \cdots & x_{1(d-1)}x_{1d} & x_{11}^2 & \cdots & x_{1d}^2 & \cdots \\ 1 & x_{21} & \cdots & x_{2d} & x_{21}x_{22} & \cdots & x_{2(d-1)}x_{2d} & x_{11}^2 & \cdots & x_{1d}^2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \cdots \\ 1 & x_{n1} & \cdots & x_{nd} & x_{n1}x_{n2} & \cdots & x_{n(d-1)}x_{nd} & x_{n1}^2 & \cdots & x_{nd}^2 & \cdots \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{p}^T(\boldsymbol{x}_1) \\ \boldsymbol{p}^T(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{p}^T(\boldsymbol{x}_n) \end{bmatrix} \end{aligned} \tag{2.3.10}$$

Here, $\boldsymbol{Y}$ is a vector containing n responses; β is a vector of p regression coefficients; ϵ is a vector of the n errors; $\boldsymbol{P}$ is an $n \times p$ model matrix, each row of which represents the expanded model form of variable settings and each column of which corresponds to one regression coefficient.

Then, by solving the minimization problem (Equation 2.3.4), the fitted regression model can be expressed by

$$\tilde{y} = \boldsymbol{P} \cdot \boldsymbol{\beta} \tag{2.3.11}$$

where

$$\boldsymbol{\beta} = \left(\boldsymbol{P}^T \boldsymbol{P}\right)^{-1} \boldsymbol{P}^T \boldsymbol{Y} \tag{2.3.12}$$

Therefore, the response of unknown point $\boldsymbol{x}_u$ can be expressed as

$$\tilde{y}\left(\boldsymbol{x}_u\right) = \boldsymbol{p}^T(\boldsymbol{x}_u)\boldsymbol{\beta} \tag{2.3.13}$$

where $\boldsymbol{p}^T(\boldsymbol{x}_u)$ is a vector representing the expanded model form for $\boldsymbol{x}_u$, which is similar to a row in $\boldsymbol{P}$.

In general, low order polynomial metamodels are used to capture the global trends of the actual model, but cannot represent a good behavior over the entire design space in many cases. Hence, it is more promising for polynomial metamodels to be used in iterative optimization procedures where successive metamodels are built in a smaller and smaller region of the design space around the proposed iterates, see more details in [16].

2.3.2 Moving least squares metamodel

Polynomial metamodels can be a good representation in small regions where the response is not complex but introduce large errors for nonlinear and multimodal responses. Moving least squares metamodels [75] take advantage of these features of polynomial metamodels and a mathematical description of of MLS metamodel can be formulated as

$$\tilde{y}\left(\boldsymbol{x}\right) = \sum_{i=1}^{p} p_i(\boldsymbol{x})\beta_i(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x}) \cdot \boldsymbol{\beta} \tag{2.3.14}$$

where $\boldsymbol{p}$ is a vector of basis functions for the metamodel and $\boldsymbol{\beta}$ is the vector containing corresponding coefficients. And p is the number of the coefficients dependent on the order of approximation.

For example, a common choice for $\boldsymbol{p}$ is linear and quadratic monomials

$$\boldsymbol{p}(\boldsymbol{x}) = \left[1, x_1, x_2, ..., x_d, x_1x_2, ..., x_ix_{i+1}, ..., \frac{x_1^2}{2}, ..., \frac{x_d^2}{2}\right]^T \tag{2.3.15}$$

For two variables, Equation 2.3.14 becomes

$$\tilde{y}(\boldsymbol{x}) = \left[1, x_1, x_2, \frac{x_1^2}{2}, x_1 x_2, \frac{x_2^2}{2}\right] \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} \tag{2.3.16}$$

The coefficients β_i are determined by the weighted least squares method, minimizing the weighted error between the response from the experimental simulation $y(\boldsymbol{x})$ and the approximated value from the metamodel $\tilde{y}(\boldsymbol{x})$

$$min \sum_{i=1}^{n} w_i \epsilon_i^2 = \min_{\beta} \sum_{i=1}^{n} w(\|\boldsymbol{x}_i - \boldsymbol{x}\|) \cdot \left[\boldsymbol{p}^T(\boldsymbol{x}_i - \boldsymbol{x}) \cdot \boldsymbol{\beta} - y(\boldsymbol{x}_i)\right]^2 \tag{2.3.17}$$

where n is the number of fitting design points, $\boldsymbol{x}_i$ is the $i-th$ input point and $w(\|\boldsymbol{x}_i - \boldsymbol{x}\|)$ are the weighting functions. The weighting functions play an significant role in MLS metamodels. Generally, $w_i \geq 0$ ensure the continuity and locality of the approximation. It is dependent on the distance between the design point $\boldsymbol{x}_i$ and the studied point $\boldsymbol{x}$. Moreover, the weight w_i reaches its maximum value at $\boldsymbol{x}_i$ and decrease within a fixed region around $\boldsymbol{x}_i$, called the domain of influence of $\boldsymbol{x}_i$. If the studied point $\boldsymbol{x}$ is outside of the influence domain at $\boldsymbol{x}_i$, the weight w_i vanishes. Hence, compared to polynomial metamodels (see Section 2.3.1), the coefficients $\boldsymbol{\beta}$ in MLS metamodel are dependent on the location of the studied point $\boldsymbol{x}$ in the design space. Thus, one polynomial fit (see Equation 2.3.14) is not valid over the complete design domain. Instead, one MLS metamodel is only valid in the surrounding areas of $\boldsymbol{x}$ where the fit is made.

By solving the problem shown by Equation 2.3.17, the coefficients $\boldsymbol{\beta}$ can be obtained as

$$\boldsymbol{\beta} = \left(\boldsymbol{P}^T \boldsymbol{W} \boldsymbol{P}\right)^{-1} \boldsymbol{P}^T \boldsymbol{W} \boldsymbol{Y} \tag{2.3.18}$$

where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}^T(\boldsymbol{x}_1 - \boldsymbol{x}) \\ \vdots \\ \boldsymbol{p}^T(\boldsymbol{x}_n - \boldsymbol{x}) \end{bmatrix}, \quad \boldsymbol{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{2.3.19}$$

$$\boldsymbol{W} = \begin{bmatrix} w(\|\boldsymbol{x}_1 - \boldsymbol{x}\|) & & 0 \\ & \ddots & \\ 0 & & w(\|\boldsymbol{x}_n - \boldsymbol{x}\|) \end{bmatrix} \tag{2.3.20}$$

Then, the MLS metamodel for the studied point $\boldsymbol{x}_u$ can be written as

$$\tilde{y}(\boldsymbol{x}_u) = \boldsymbol{p}^T(\boldsymbol{x}_u) \cdot \boldsymbol{\beta} \tag{2.3.21}$$

$$= \boldsymbol{p}^T(\boldsymbol{x}_u) \cdot \left(\boldsymbol{P}^T(\boldsymbol{x}_u)\boldsymbol{W}(\boldsymbol{x}_u)\boldsymbol{P}(\boldsymbol{x}_u)\right)^{-1} \boldsymbol{P}^T(\boldsymbol{x}_u)\boldsymbol{W}(\boldsymbol{x}_u)\boldsymbol{Y} \tag{2.3.22}$$

Note that because $\boldsymbol{\beta}$ is a function of $\boldsymbol{x}_u$, a new MLS model is required to be fitted for every new point of $\boldsymbol{x}_u$. Moreover, in order to construct the MLS metamodel, there should be enough fitting points within the domain of influence. It can be done by changing the weight functions or adjusting the radius of the domain of influence. Usually, the denser the design space is scattered, the smaller the domain of influence should be, and the more accurate the metamodel becomes.

2.3.3 Kriging metamodel

Kriging metamodel was firstly developed by Georges Matheron [125] and has gained various improvements for tackling complex engineering optimization problems[26, 74, 126]. The basic theory behind Kriging is that the deterministic response $y(\boldsymbol{x})$ of computer model can be represented as

$$y(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x}) \cdot \boldsymbol{\beta} + Z(\boldsymbol{x}) \tag{2.3.23}$$

where $\boldsymbol{p}^T(\boldsymbol{x}) \cdot \boldsymbol{\beta}$ is a polynomial regression model which is similar to Equation 2.3.14 and $Z(\boldsymbol{x})$ is the random process. This stochastic process $Z(\boldsymbol{x})$ has mean 0, variance σ^2 as

$$E[Z(\boldsymbol{x})] = 0 \tag{2.3.24}$$

$$E[Z(\boldsymbol{x})Z(\boldsymbol{w})] = \sigma^2 R(\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{x}) \tag{2.3.25}$$

where $\boldsymbol{\theta}$ is a correlation coefficient vector and $R(\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{x})$ is a Gaussian spatial correlation function [127], which can be described as

$$R(\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{x}) = \prod_{i=1}^{d} R_i(\theta_i, w_i, x_i) \tag{2.3.26}$$

Two commonly used functions are the exponential and the Gaussian correlation functions, i.e.,

$$R(\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{x}) = \prod_{i=1}^{d} e^{-\theta_i |w_i - x_i|} \tag{2.3.27}$$

and

$$R(\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{x}) = \prod_{i=1}^{d} e^{-\theta_i |w_i - x_i|^2} \tag{2.3.28}$$

respectively. $|w_i - x_i|$ is the distance between $\boldsymbol{w}$ and $\boldsymbol{x}$ point in $i-th$ dimension, d is the number of variables, and θ_i is the correlation parameter for $i-th$ variable. In general, θ_i is essentially a width measure which indicates how far the influence of a fitting point extends [128]. A low θ_i suggests that all designs will have a high correlation R with similar $Z(x_i)$. In contrast, a high θ_i indicates that the random process for the $i-th$ variable $Z(x_i)$ is significantly different among sample points. Therefore, the elements of $\boldsymbol{\theta}$ provide an insight into the importance of different variables, which is helpful for researchers to select or scale the design variables if necessary.

In order to build a Kriging metamodel involving n sampling points, both the polynomial coefficients $\boldsymbol{\beta}$ as well as the correlation coefficients $\boldsymbol{\theta}$ have to be figured out. Similar to polynomial metamodel (Section 2.3.1) and moving least squares metamodel (Section 2.3.2), the regression coefficients $\boldsymbol{\beta}$ can be obtained by solving the regression problem

$$\boldsymbol{P\beta} \approx \boldsymbol{Y} \tag{2.3.29}$$

As defined in Equation 2.3.9 and 2.3.10, $\boldsymbol{P}$ is the model matrix and $\boldsymbol{Y}$ is a vector of the n observed responses. Hence, a general solution of $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta} = \left(\boldsymbol{P}^T \boldsymbol{R}^{-1} \boldsymbol{P}\right)^{-1} \boldsymbol{P}^T \boldsymbol{R}^{-1} \boldsymbol{Y} \tag{2.3.30}$$

where

$$\boldsymbol{R} = \begin{bmatrix} R(\boldsymbol{\theta}, \boldsymbol{x}_1, \boldsymbol{x}_1), \cdots, R(\boldsymbol{\theta}, \boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots, \ddots, \vdots \\ R(\boldsymbol{\theta}, \boldsymbol{x}_n, \boldsymbol{x}_1), \cdots, R(\boldsymbol{\theta}, \boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} \tag{2.3.31}$$

And $\boldsymbol{\theta}$ can be determined by solving the nonlinear optimization problem, i.e., maximizing the log-likelihood function

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & L(\boldsymbol{\theta}) = -\frac{1}{2}\left[n \ln(\tilde{\sigma}^2) + \ln|\boldsymbol{R}|\right] \\ s.t. \quad & \theta_i > 0, i = 1, ..., d \end{aligned} \tag{2.3.32}$$

where $|\boldsymbol{R}|$ is the determinant of $\boldsymbol{R}$. The estimation of the variance can be given by

$$\tilde{\sigma}^2 = \frac{(\boldsymbol{Y} - \boldsymbol{P}\boldsymbol{\beta})^T \boldsymbol{R}^{-1} (\boldsymbol{Y} - \boldsymbol{P}\boldsymbol{\beta})}{n} \tag{2.3.33}$$

When $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are determined, the predicted response of an unknown point $\boldsymbol{x}_u$ can be calculated by

$$\tilde{y}(\boldsymbol{x}_u) = \boldsymbol{p}^T(\boldsymbol{x}_u)\boldsymbol{\beta} + \boldsymbol{r}^T(\boldsymbol{x}_u)\boldsymbol{R}^{-1} (\boldsymbol{Y} - \boldsymbol{P}\boldsymbol{\beta}) \tag{2.3.34}$$

where $\boldsymbol{p}^T(\boldsymbol{x}_u)$ is a vector containing polynomials like Equation 2.3.15, $\boldsymbol{\beta}$ is the estimated regression parameters, $\boldsymbol{r}^T(\boldsymbol{x}_u) = [R(\boldsymbol{\theta}, \boldsymbol{x}_u, \boldsymbol{x}_1), R(\boldsymbol{\theta}, \boldsymbol{x}_u, \boldsymbol{x}_2), ..., R(\boldsymbol{\theta}, \boldsymbol{x}_u, \boldsymbol{x}_n)]^T$ is a vector of correlation functions between $\boldsymbol{x}_u$ and n sample points, $\boldsymbol{R}$ is the matrix of correlation functions for all sample points and $\boldsymbol{Y}$ is a vector of the corresponding responses of fitting points.

It should be noted here that the maximization optimization problem (Equation 2.3.32) requires large computational expense, which limits the application of Kriging metamodel to problems of low dimensionality, with d usually limited to around 20 [128]. But the Kriging metamodel is flexible due to the properties of various correlation functions, which is suitable for approximating the entire design space [9]. In addition, as in many cases we do not have a priori knowledge of the patterns or trends in the data, ordinary Kriging where $\boldsymbol{p}^T(\boldsymbol{x}) \cdot \boldsymbol{\beta}$ is set to be a constant are much popular in actual engineering applications.

2.3.4 Radial basis function metamodel

Radial basis function(RBF) metamodel was first presented by Roland Hardy [129] and has gained popularity for interpolating multi-dimensional data. Radial basis functions can be of many forms but are always radially symmetric because they are dependent on the radial distance from a specific point $\boldsymbol{x}_i$ as

$$\phi(\boldsymbol{x}, \boldsymbol{x}_i) = \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) = \phi(r) \tag{2.3.35}$$

where r is the distance between the point $\boldsymbol{x}$ and $\boldsymbol{x}_i$. Commonly used RBFs are multiquadric, Gaussian, thin plate spine and cubic as shown in Equation 2.3.36, where c is a shape parameter that affects the smoothness of the

function.

$$\phi(r) = \begin{cases} e^{-\left(\frac{r}{c}\right)^2} & \text{Gaussian} \\ \sqrt{r^2 + c^2} & \text{Multiquadric} \\ r^2 \ln r & \text{Thin plate spline} \\ r^3 & \text{Cubic} \end{cases} \tag{2.3.36}$$

A RBF metamodel can be written as

$$\tilde{y} = s(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \cdot \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) = \boldsymbol{w}^T \boldsymbol{\phi} \tag{2.3.37}$$

where $\boldsymbol{w}$ is the weighting coefficients associated with each sample point $\boldsymbol{x}_i$ and $\boldsymbol{\phi}$ is a vector containing the evaluations of the radial basis function for distances between the unknown point $\boldsymbol{x}$ and different sample points $\boldsymbol{x}_i$. Therefore, this formulation leads to an interpolation problem involving n input points as

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{w} \tag{2.3.38}$$

where $\boldsymbol{Y}$ is the known responses, $\boldsymbol{w}$ is the unknown coefficients and $\boldsymbol{B}$ is the $n \times n$ symmetric interpolation matrix where $B_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$. In this way, the number of RBFs is equal to the number of sample points. Hence, $\boldsymbol{w} = \boldsymbol{B}^{-1}\boldsymbol{Y}$ if $\boldsymbol{B}$ is a symmetric positive definite matrix[128]. Note that the shape parameter c in Gaussian and multiquadric RBFs has significant effects on the conditioning of the problem. When $c \to \inf$, the elements of $\boldsymbol{B}$ approach constant values and the linear system in Equation 2.3.38 is ill-conditioned. Generally, a large value of c gives a wider affected region, i.e., points further away from the studied point $\boldsymbol{x}$ are able to affect the prediction of the response $s(\boldsymbol{x})$. On the other hand, a small value of c means that only surrounding points of $\boldsymbol{x}$ can influence the prediction.

A Kriging metamodel can be seen as a special case of a RBF metamodel combined with an additional low order polynomial. In fact, if the polynomial term is 0 and θ_i = constant in Equation 2.3.23, a Kriging metamodel with Gaussian correlation functions is exactly a RBF metamodel with Gaussian basis functions.

2.4 Optimization methods

Optimization algorithms can be classified into two broad categories: local or global optimization algorithms. A local optimization algorithm attempts to seek a local optimum, i.e., there is no guarantee that this optimum is also

the global one. Most local optimization algorithms are gradient-based, i.e., the optimization process is dependent on the gradient information to find an optimal solution [130]. Typically, these techniques prove to be efficient, can solve large-scale problems, and usually require little parameter tuning. On the other hand, besides of only obtaining local optima, they are not suitable for solving discrete optimization problems (in which at least one design variable takes discrete values) and are probably sensitive to numerical noise.

In contrast, a global optimization algorithm focuses on finding the global or near global optimum. There are two main categories of global optimization methods: deterministic methods and heuristic methods [69]. Deterministic methods behave predictable because given the same input, the algorithm will follow the same sequence of states and output the same result in each iteration. In general, deterministic methods can only be successfully used when the optimization problem has certain mathematical characteristics that usually do not exist. For more details, readers are advised to refer to [2].

The heuristic methods are typically inspired by some phenomenon from nature, and prove to be robust and well suited for discrete optimization problems. As described in [69], they usually do not require any derivative information and can search large design spaces. On the other hand, they often require many times more objective function evaluations than deterministic methods and are poor in handling constraints [130]. Popular heuristic optimization algorithms include genetic algorithms [4], ant colony optimization (ACO) [6], particle swarm optimization (PSO) [5], simulated annealing [131] and so on.

In subsequent sections, a popular gradient-based algorithm - sequential quadratic programming (SQP) and particle swarm optimization will be presented in details.

2.4.1 Sequential quadratic programming (SQP)

Sequential quadratic programming (SQP) is one of the most powerful methods for engineering optimization applications. It is established on a profound theoretical foundation and aims at providing the numerical solution of constrained nonlinear optimization problems (NLP).

We consider a typical NLP is formulated as

$$\begin{aligned} min \quad & f(\boldsymbol{x}) \\ s.t. \quad & \boldsymbol{h}(\boldsymbol{x}) = 0 \\ & \boldsymbol{g}(\boldsymbol{x}) \leq 0 \\ where \quad & \boldsymbol{x} \in \mathbb{R}^d \end{aligned} \tag{2.4.1}$$

where $f : \mathbb{R}^d \mapsto \mathbb{R}$ is the objective function, the functions $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{R}^p$ and $\boldsymbol{g} : \mathbb{R}^d \mapsto \mathbb{R}^m$ are the equality and inequality constraints respectively.

SQP is an iterative procedure which approximates the NLP (Equation 2.4.1) by a sequence of quadratic programming (QP) subproblems, in which the solutions can thus be obtained easily. Specifically, the Lagrangian functional $\mathcal{L} : \mathbb{R}^{d \times p \times m \mapsto \mathbb{R}}$ associated with the NLP can be defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{x}) \tag{2.4.2}$$

where the vectors $\boldsymbol{\lambda} \in \mathbb{R}^p$ and $\boldsymbol{\mu} \in \mathbb{R}^m_+$ are Lagrangian multipliers. The solution $\boldsymbol{x}^* \in \mathbb{R}^d$ of the NLP should satisfy the Karush-Kuhn-Tucker (KKT) conditions, i.e.,

$$\begin{aligned} \nabla f(\boldsymbol{x}^*) + \boldsymbol{\lambda}^* \nabla \boldsymbol{h}(\boldsymbol{x}^*) + \boldsymbol{\mu}^* \nabla \boldsymbol{g}(\boldsymbol{x}^*) &= 0 \\ \boldsymbol{h}(\boldsymbol{x}^*) &= 0 \\ \boldsymbol{g}(\boldsymbol{x}^*) &\leq 0 \\ \boldsymbol{\mu}^* \boldsymbol{g}(\boldsymbol{x}^*) &= 0 \end{aligned} \tag{2.4.3}$$

In order to solve Equation 2.4.3, a sequence of QP subproblems that reflects the local properties of the NLP have to be constructed and solved in each iteration step. Using Taylor series expansion, the auxiliary QP problem with regard to the current iterate $\boldsymbol{x}^k$ can be written as

$$\begin{aligned} min \quad & \nabla f(\boldsymbol{x}^k)^T d(\boldsymbol{x}) + \frac{1}{2} d(\boldsymbol{x})^T B_k d(\boldsymbol{x}) \\ s.t. \quad & \boldsymbol{h}(\boldsymbol{x}^k) + \nabla \boldsymbol{h}(\boldsymbol{x}^k)^T d(\boldsymbol{x}) = 0 \\ & \boldsymbol{g}(\boldsymbol{x}^k) + \nabla \boldsymbol{g}(\boldsymbol{x}^k)^T d(\boldsymbol{x}) \leq 0 \\ where \quad & d(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}^k \\ & B_k = \boldsymbol{H} f(\boldsymbol{x}^k) \end{aligned} \tag{2.4.4}$$

where $\boldsymbol{H} f(\boldsymbol{x}^k)$ is the Hessian of $f(\boldsymbol{x}^k)$. As a result, instead of solving a nonlinear optimization problem directly, in each iteration only a quadratic subproblem should be solved. For more details of SQP, please refer to [2].

2.4.2 Particle swarm optimization (PSO)

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995 [5], based on swarm behavior observed in nature such as fish and bird schooling. Since then, PSO has attracted a lot of attention, and now becoming a main representative form of swarm intelligence. PSO has been applied to almost every area in optimization, computational intelligence, and design/scheduling applications. There are at least two dozens of PSO variants, as well as hybrid algorithms obtained by combining PSO with other existing algorithms, which are also increasingly popular [132].

In PSO, each particle represents a point in the design space of the optimization problem with an associated velocity vector. In each iteration of PSO, the velocity vector is updated by using a linear combination of three terms:

$$\boldsymbol{V}_i^{k+1} = \omega \boldsymbol{V}_i^k + \alpha \boldsymbol{\epsilon}_1 \left(\boldsymbol{pbest}_i^k - \boldsymbol{x}_i^k \right) + \beta \boldsymbol{\epsilon}_2 \left(\boldsymbol{gbest}_i^k - \boldsymbol{x}_i^k \right) \tag{2.4.5}$$

where ω is the parameter called inertial weight, k is the iteration parameter, i is the index of each particle, α and β are parameters called acceleration coefficients, ϵ_1 and ϵ_2 are two homogeneously distributed random vectors generated within the interval $[0, 1)$ respectively. In the right-hand side of Equation 2.4.5, the first term called inertia or momentum works as a memory of the previous flight direction, preventing the particle from changing direction thoroughly. The second term, called the cognitive component describes the tendency of particles return to the previously found best positions. The last term, called the social component quantifies the group norm or standard that should be attained. In other words, each particle tends to move toward to the current global best position $\boldsymbol{g}_{best}$ and personal best known location $\boldsymbol{p}_{best}$, while moving randomly in the meantime [133]. After the velocity vector of each particle is determined, the particle i will move from $\boldsymbol{x}_i^k$ to $\boldsymbol{x}_i^{k+1}$ by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{V}_i^{t+1} \tag{2.4.6}$$

In this way, particles are able to search in the entire design space until the global best position $\boldsymbol{g}_{best}$ no longer improves or after a certain number of iterations [134].

Although PSO has gained huge popularity for its easy-implementation and easy-maintenance characteristics, its convergence in a mathematics sense is still unknown. In addition, it has difficulties on handling complex constraints and balancing the exploration and exploitation abilities.

2.5 Research gaps

In summary, there is lack of research in the area of efficient constrained black-box optimization. Direct methods which depend exclusively on simulations for optimization are impractical in solving real-world optimization problems due to the unbearable number of function evaluations. Metamodel technique is a promising way for reducing the computational overhead because it can replace the costly black-box model with simple analytical expression. This kind of approach has been widely researched but most of them are just applied to solve unconstrained problems. Moreover, the strategy of selecting the sampling points for fitting metamodels is simple. The majority of the methods randomly distribute points across the global design space and use these points for building metamodels. It lacks systematic theory in weighing different points which are in various optimization conditions. Furthermore, most of the algorithms aim to updating the global metamodel which is valid in the global design space through a lot of approaches. But it is not a feasible strategy when the implicit black-box model is very complex. Few researches focus on developing the sequential approximation strategy where the metamodel is only valid in local region because it is highly difficult to guarantee the global search ability. Trust-region based searching scheme [135] seems a promising sequential approximation strategy which decomposes the original optimization problem into a sequence of approximate subproblems in a series of trust regions. But how to move and resize the trust region to facilitate the algorithm to converge to the global solution is difficult and this area lacks improvement during the past ten years. Last but not the least, although some researchers stated that their proposed methods could deal with constrained black-box optimization problems, only low-dimensional problems were tested and analyzed. Relatively few algorithms could handle expensive and high-dimensional black-box optimization problems under severely limited budget. Hence, this thesis aims to propose a versatile optimization framework for tackling expensive, large-scale and severely constrained black-box optimization problems with high accuracy, robustness and efficiency.

3

Intrinsically linear function assisted and trust region based optimization method

In this chapter, an intrinsically linear function assisted and trust region based optimization method (IATRO) is developed for solving expensive CBO problems. It adopts strong points of the multipoint approximation method (MAM) developed by Toropov et al. [135]. For better understanding of the complete development process of IATRO, a brief overview of MAM is presented in Section 3.1. The main theories behind MAM is also the fundamentals of IATRO, which are given from Section 3.2 to Section 3.4. Following the framework of MAM, IATRO is a fully reimplementation of MAM in Python with various improvements. The details are shown in Section 3.5. Next in Section 3.6, to validate the performance of IATRO, 26 benchmark problems are tested and the results of IATRO are compared with some good results obtained by several state-of-the-art metaheuristic algorithms and metamodel-based algorithms. And a summary is given in Section 3.7.

3.1 Outline of MAM

MAM is a metamodel-based optimization method that applies a sequential strategy with domain reduction. In each iteration of MAM, only a local metamodel valid in the current search subregion is built. Moreover, the subregion where the fitting points are generated, called the region of interest (also called the trust region), is reduced in size and moved within the design space based on the trust region strategy. In this way, MAM replaces the original optimization problem (Equation 2.1.1) by a succession of

approximate subproblems as

$$\begin{aligned} \underset{\boldsymbol{x}\in Q_k}{min} \quad & \tilde{f}^k(\boldsymbol{x}) \\ s.t. \quad & \tilde{g}_j^k(\boldsymbol{x})\,(j=1,...,m) \\ where \quad & Q^k=[\boldsymbol{A}^k,\boldsymbol{B}^k]\subseteq\mathbb{R}^d, \\ & A_i^k\geq A_i,\ B_i^k\leq B_i\,(i=1,...,d) \end{aligned} \tag{3.1.1}$$

where $\boldsymbol{x}$ is the vector of design variables, $\tilde{f}^k(\boldsymbol{x})$ and $\tilde{g}_j^k(\boldsymbol{x})\,(j=1,...,m)$ are approximated objective and constrained functions respectively, Q^k is the subregion in k_{th} iteration that bounded by $\boldsymbol{A}^k$ and $\boldsymbol{B}^k$. The solution of the subproblem becomes the starting point for the next iteration, the trust region is modified and the metamodel is updated until certain termination criteria are satisfied. The main processes of MAM are shown in Fig. 3.1 and the details are described in subsequent sections.

3.2 Design of experiments (DOE)

In order to build a metamodel, a dataset of inputs (a vector set of design variables) and corresponding outputs (response values) are required. Design of experiments (DOE) is applied to determine the distribution of experimental points in the design space in order to get the best possible information from a limited sample size. Toropov et al. [135] proposed to apply two following strategies.

3.2.1 Maxmin stochastic sampling (MSS)

For the purpose of improving the uniformity of a stochastic sampling design, Toropov et al. [135] proposed to impose an additional constraint (a constraint on the minimal distance between any two points) on a random DOE, which can be expressed by

$$\frac{Dist}{Diag}\geq r \tag{3.2.1}$$

where

$$Diag=\sqrt{\sum_{i=1}^{d}(B_i^k-A_i^k)^2} \tag{3.2.2}$$

$$Dist=\sqrt{\sum_{i=1}^{d}(x_i^e-x_i^p)^2}\quad(p=1,...,P) \tag{3.2.3}$$

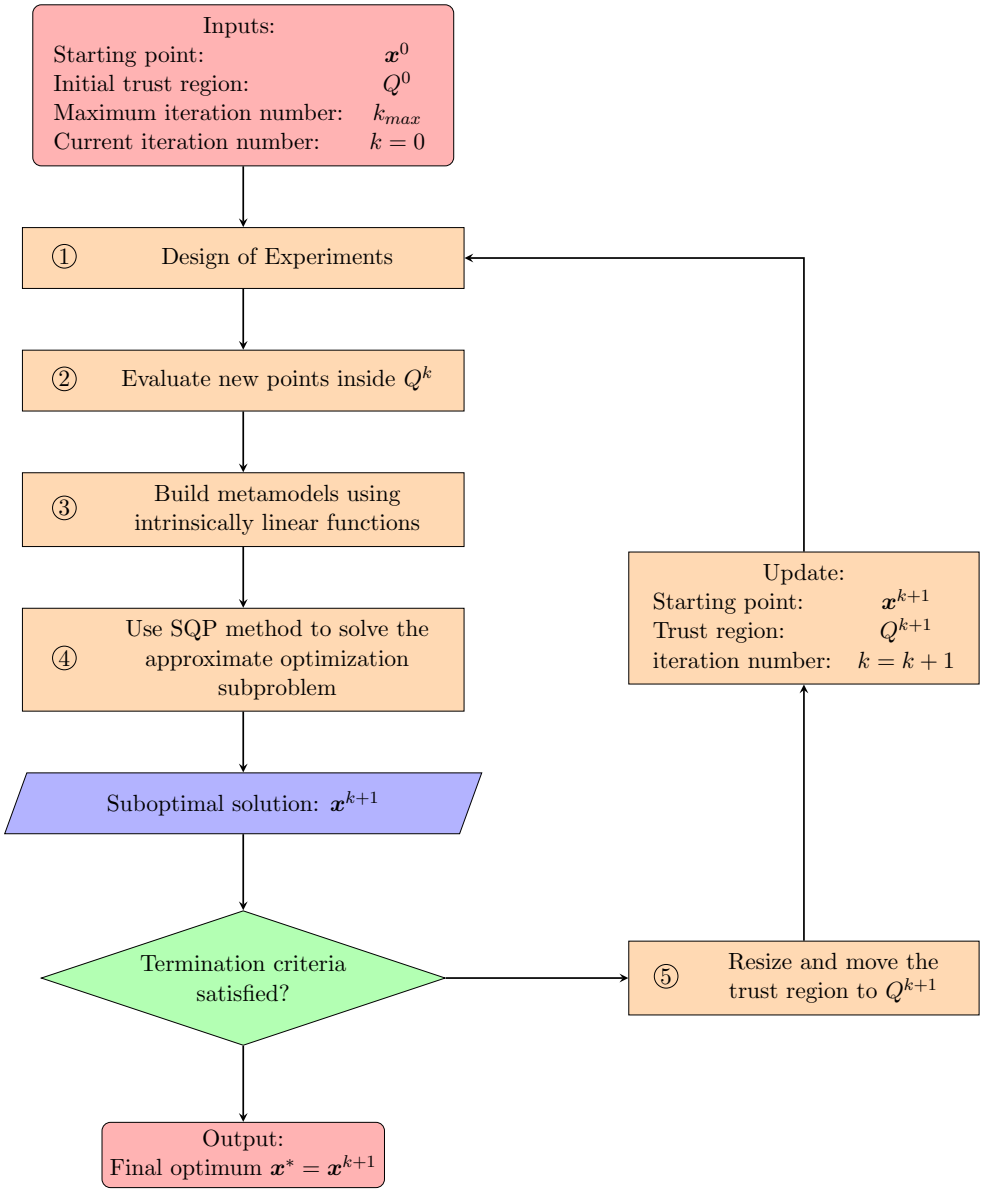


Figure 3.1: Flow chart of the Multipoint Approximation Method

In Equation 3.2.1-3.2.2, *Diag* represents the characteristic size of the k_{th} trust region (i.e. L^2 distance), $\boldsymbol{x}^e$ is a new sampling point to be generated, $\boldsymbol{x}^p$ $(p = 1, ..., P)$ are P previously generated points in the k_{th} trust region, and r is a threshold ratio controlling the uniformity of the scattered points. r is initially set to 0.95 and if the randomly generated sampling point $\boldsymbol{x}^e$ can not satisfy the condition (Equation 3.2.1) after a user-specified number of times, r will be reduced iteratively using

$$r = r \times 0.9 \tag{3.2.4}$$

until the constraint (Equation 3.2.1) is satisfied. The pseudocode of this maxmin stochastic sampling (MSS) approach is given in Alg. 1 for reference.

Algorithm 1: Maxmin Stochastic Sampling (MSS)

Input:

- $\boldsymbol{x}^k$: Starting point in k_{th} iteration.
- n_s: The number of required sampling points.
- $\boldsymbol{A}^k$: Lower bounds of the current search subregion.
- $\boldsymbol{B}^k$: Upper bounds of the current search subregion.

Output:

The set of sampling points X_{MSS}.

Function `MSS(`$\boldsymbol{x}^k$`,`n_s`,`$\boldsymbol{A}^k$`,`$\boldsymbol{B}^k$`):` ▷ `Maxmin Stochastic Sampling`

Put $\boldsymbol{x}^k$ in the sampling pool X_{MSS}.
$Diag = \|\boldsymbol{A}^k - \boldsymbol{B}^k\|_2$.
for i **in** $[0, n_s - 1]$ **do**
Generate a point $\boldsymbol{x}$ satisfied:
$\frac{Dist}{Diag} \geq r$. ▷ `The initial value of` r `is 0.9`
▷ $Dist$ `is the minimum distance between` $\boldsymbol{x}$ `and points in` X_{MSS}.
▷ r `will be reduced after a certain number of generations.`
Put $\boldsymbol{x}$ in the sampling pool X_{MSS}.
Return X_{MSS}

3.2.2 Extended box selection (EBS)

During the progression of the optimization procedure, a database that contains various designs with corresponding response function values becomes available. In order to improve the quality of approximations in the current trust region, the extended box selection (EBS) strategy is employed as described in Algorithm 2. The main idea behind EBS is to reutilize the previous information, i.e., the evaluated designs in the previous iteration. Generally, points located far from the current trust region would not bring improvement in the quality of the resulting metamodel. Therefore, Toropov [135] suggested that points located in the 'neighborhood' of the current trust region should also be included for building the metamodel. The 'neighborhood' is a box surrounding the current search subregion as depicted in Fig. 3.2. And this extended box is related to the current trust region as

$$B_i^{k,ext} - A_i^{k,ext} = \Delta ext \left(B_i^k - A_i^k \right) \tag{3.2.5}$$

where Δext is typically between 1.4 and 2.0.

3.3 Intrinsically linear function regression (ILFR)

The process of metamodel building in MAM can be described as an assembly of multiple surrogates into one single metamodel using linear regression.

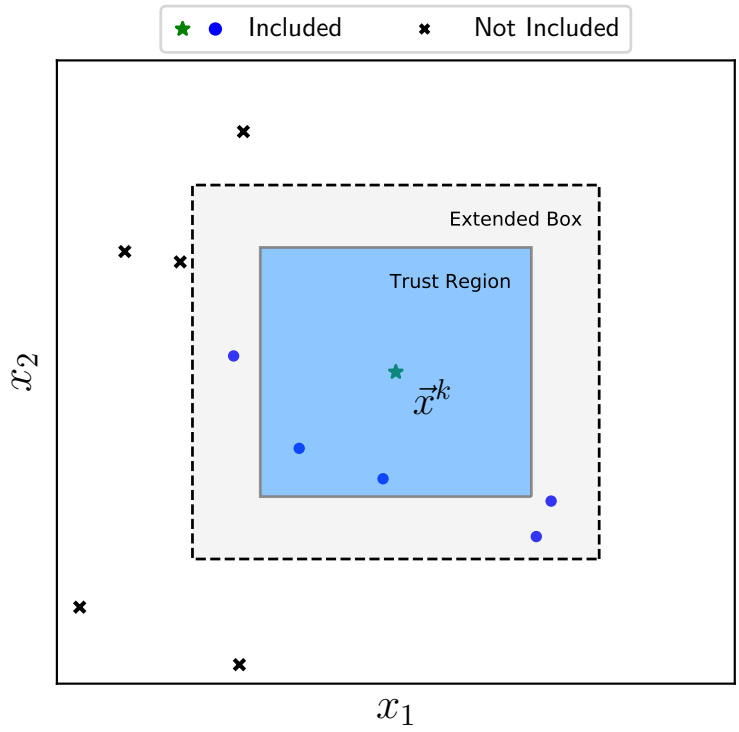


Figure 3.2: Selection of response evaluations carried out during an optimization trial. Only the designs located in the neighborhood of the current trust region are included in the weighted least squares fitting

Algorithm 2: Extended-box Selection (EBS)

Function EBS($\boldsymbol{x}^k, X_{all}, \Delta_{ext}, N_{plan}, \boldsymbol{A}^k, \boldsymbol{B}^k$):

Input:

- $\boldsymbol{x}^k$: Starting point in k_{th} iteration.
- X_{all}: The database of all sampling points.
- Δ_{ext}: The relative size of the extended box.
- N_{plan}: The default number of required sampling points.
- $\boldsymbol{A}^k$: Lower bounds of the current search subregion.
- $\boldsymbol{B}^k$: Upper bounds of the current search subregion.

Output:

- n_{ext}: The number of points located in the extended search subregion.
- Q_{ext}: The extended search subregion.
- X_{EBS}: The set of points selected.

▷ Define the extended box Q_{ext}.

$Q_{ext}: \ B_i^{k,ext} - A_i^{k,ext} = \Delta_{ext} \cdot (B_i^k - A_i^k)$.

▷ Define the counter to record how many points have been selected.

$n_{ext} = 0$.

for $\boldsymbol{x}_i$ **in** X_{all} **do**

  if $\boldsymbol{x}_i \in Q_{ext}$ **then** ▷ Choose points located in Q_{ext}

    Put $\boldsymbol{x}_i$ in the pool X_{EBS}.

    $n_{ext} = n_{ext} + 1$. ▷ Increment the counter

Return $n_{ext}, Q_{ext}, X_{EBS}$

Therefore, there are two stages of metamodel building.

In the first stage, the parameters $\boldsymbol{a}_l$ of a individual surrogate φ_l is determined by solving a weighted least squares problem involving n fitting points as

$$min \quad \sum_{i=1}^{n} \omega_i \left[F(\boldsymbol{x}_i) - \varphi_l(\boldsymbol{x}_i, \boldsymbol{a}_l)\right]^2 \tag{3.3.1}$$

where ω_i denote the weighting parameters, i.e., the inequality of data obtained at different sampling points, F is the original function needs to be approximated. It should be noted here that in MAM, both the objective and constraint functions will be approximated by Equation 3.3.1. The simplest case of φ_l is the first order polynomial metamodel and more complex ones are intrinsically linear functions (ILF) [136] that have been successfully applied for solving various design optimization problem [78–80, 135, 137]. ILF are nonlinear but they can be led to linear ones by simple transformations. Currently, five functions are considered in the regressor pool $\{\varphi_l(\boldsymbol{x})\}$ as

$$\begin{aligned}
\varphi_1(\boldsymbol{x}) &= a_0 + \sum_{i=1}^{d} a_i x_i \\
\varphi_2(\boldsymbol{x}) &= a_0 + \sum_{i=1}^{d} a_i x_i^2 \\
\varphi_3(\boldsymbol{x}) &= a_0 + \sum_{i=1}^{d} a_i / x_i \\
\varphi_4(\boldsymbol{x}) &= a_0 + \sum_{i=1}^{d} a_i / x_i^2 \\
\varphi_5(\boldsymbol{x}) &= a_0 \prod_{i=1}^{d} x_i^{a_i}
\end{aligned} \tag{3.3.2}$$

In Equation 3.3.1, Toropov [77] claimed that the selection of weighting factors ω_i should reflect (i) the quality of the objective function and (ii) the location of a design point with respect to the border between the feasible and the infeasible design subspace. Therefore, ω_i are defined as

$$w_i = w_i^o \cdot w_i^c \tag{3.3.3}$$

$$w_i^o = [\frac{f(\boldsymbol{x}^k)}{f(\boldsymbol{x}_i)}]^\beta \tag{3.3.4}$$

$$w_i^c = \begin{cases} 1 & \text{for objective } f(x) \\ [g(\boldsymbol{x}) + 1]^\alpha & \text{if } g(\boldsymbol{x}) \leq 0 \\ [g(\boldsymbol{x}) + 1]^{-\alpha} & \text{if } g(\boldsymbol{x}) \geq 0 \end{cases} \tag{3.3.5}$$

where $\alpha, \beta > 0$ are user defined constants, here $\alpha = 4, \beta = 1.5$ are used; $\boldsymbol{x}^k$ is the starting point in k_{th} iteration and $\boldsymbol{x}_i$ is the i_{th} design point in the fitting points. With this definition, a point with a larger objective function has a smaller weighting coefficient component w_i^o. For a constraint function $g(\boldsymbol{x})$, a point which is much closer to the boundary of the feasible region of $g(\boldsymbol{x})$, is given a larger weighting coefficient component w_i^c. For building a surrogate of the objective function $f(\boldsymbol{x})$, the weighting coefficient w_i will only consider the component w_i^o. But for building a surrogate of the constraint function $g(\boldsymbol{x})$, the weighting coefficient w_i will also take the constraint component w_i^c into consideration.

In the second stage, for each function ($f(\boldsymbol{x})$ or $g(\boldsymbol{x})$), different surrogates are assembled into one metamodel as

$$\tilde{F}(\boldsymbol{x}) = \sum_{l=1}^{n_l} b_l \varphi_l(\boldsymbol{x}) \tag{3.3.6}$$

where n_l is the number of surrogates applied in the model bank $\{\varphi_l(\boldsymbol{x})\}$, and b_l is the regression coefficient corresponding to each surrogate $\varphi_l(\boldsymbol{x})$, which reflects the quality of the individual $\varphi_l(\boldsymbol{x})$ on the set of validation points. Similar to Equation 3.3.1, b_l can be determined in the same manner as

$$min \quad \sum_{i=1}^{n} \omega_i \left[F(\boldsymbol{x}_i) - \tilde{F}(\boldsymbol{x}_i, \boldsymbol{b}) \right]^2 \tag{3.3.7}$$

It should be noted that in the process of metamodel building, the DOE is fixed, i.e., ω_i remains unchanged across the aforementioned stages.

3.4 Trust region strategy

Once the metamodels for both objective and constraint functions have been established, the solution of the subproblem (Equation 3.1.1) can be found by the sequential quadratic programming (SQP) method described in Section 2.4.1. Then, a new search subregion including its dimensions and locations must be specified in the next iteration for seeking the optimum. To achieve this goal, several indicators have been formulated to help enhance the search capability as follows:

The first indicator is to evaluate the quality of the metamodel and focused on the accuracy of the constraint approximations at the obtained sub-optimal

point $\boldsymbol{x}^{k+1}$. This is based on the following equation:

$$E^k = Max\left(|\frac{\tilde{g}\left(\boldsymbol{x}^{k+1}\right) - g\left(\boldsymbol{x}^{k+1}\right)}{g\left(\boldsymbol{x}^{k+1}\right)}|\right) \tag{3.4.1}$$

where $\tilde{g}\left(\boldsymbol{x}^{k+1}\right)$ and $g\left(\boldsymbol{x}^{k+1}\right)$ are normalized functions of the approximate and true constraints at the sub-optimal point $\boldsymbol{x}^{k+1}$, respectively. In this way, a single maximal error quantity between explicit approximation and implicit simulation is defined. Then, the quality of metamodel can be labeled as 'bad', 'reasonable' or 'good' shown below.

$$E^k \Rightarrow \begin{cases} \geq 0.25 \cdot S^k & \Rightarrow \text{'Bad'} \\ \leq 0.01 \cdot S^k & \Rightarrow \text{'Good'} \\ Else & \Rightarrow \text{'Reasonable'} \end{cases} \tag{3.4.2}$$

where S^k represents the maximum ratio of the dimension length between the present trust region and the entire design space, defined by

$$S^k = Max\left(\frac{B_i^k - A_i^k}{B_i - A_i}\right) \quad (i = 1, ..., d) \tag{3.4.3}$$

The second indicator is to indicate the location of the current iterate $\boldsymbol{x}^{k+1}$ in the present search subregion. For each dimension, if none of the current move limits $(\boldsymbol{A}^k, \boldsymbol{B}^k)$ is active, this solution is regarded as 'Internal', otherwise it is viewed as 'External'.

The third and fourth indicator reflects the movement history for the entire optimization process. For this purpose, the angle between the last two move vectors is calculated. The formulation of this measure θ^k is given below:

$$\theta^k = \frac{\boldsymbol{x}^{k+1} - \boldsymbol{x}^k}{\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|} \cdot \frac{\boldsymbol{x}^k - \boldsymbol{x}^{k-1}}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|} \tag{3.4.4}$$

If $\theta^k > 0$ holds, the movement will be denoted as 'Forward', while $\theta^k \leq 0$ is denoted as moving 'Backward'. Moreover, if $\theta^k \leq 0.3$, the convergence history is labelled as 'Curved', otherwise 'Straight'.

The fifth indicator in MAM, as a termination criterion, is the size of the current search subregion. It can be marked as 'Small' or 'Large' according to the quality of the metamodel determined by the first indicator. When the approximations are 'Bad' and $S^k \leq 0.005$, the present search subregion is considered 'Small'. When the approximations are 'Reasonable' or 'Good', the trust region is denoted as 'Small' if $S^k \leq 0.1$. However, this 'Small'

Table 3.1: Schematic description of trust region strategy in MAM

Approximation: 'Bad'			
'Small'	'Large'		
Stop: No Convergence Found	Straight Reduce ($\tau = 0.8$)	Curved Reduce ($\tau = 1.5$)	

Approximation: 'Reasonable' or 'Good'					
'Internal'			'External'		
'Small'	'Large'		'Backward'	'Forward'	
Stop: Convergence Found	'Close' Reduce ($\tau = 4$)	'Far' Reduce ($\tau = 2$)	Reduce ($\tau = 1.5$)	'Straight' Enlarge ($\tau = 0.8$)	'Curved' Keep ($\tau = 1$)

trust region is actually too big for well fitting the original functions. MAM will either obtain an infeasible solution or obtain a solution that far from the optimum because of the low-fidelity approximations in this 'Small' region. Thus, a modification of the fifth indicator has been made in IATRO. It still indicates the size of the current search subregion but it is not dependent on the quality of the approximations now. The size of the trust region is deemed as 'Small' if $S^k \leq \Delta_{min}$ where Δ_{min} is a user-specified coefficient representing the minimum relative size of the trust region. Otherwise if $S^k > \Delta_{min}$, the trust region is marked as 'Large'.

The sixth indicator is based on the most active constraint. It is considered to be 'Close' to the boundary between the feasible and infeasible design space if $\boldsymbol{g}_{max}(\boldsymbol{x}^{k+1}) \in [-0.1, 0.1]$, otherwise it is denoted as 'Far'.

Both reduction and enlargement of the trust region is executed using

$$B_i^{k+1} - A_i^{k+1} = \frac{1}{\tau}\left(B_i^k - A_i^k\right) \ (i = 1, ..., d) \tag{3.4.5}$$

where τ is the resizing parameter.

Table 3.1 summarizes the strategy applied for moving and resizing the trust region based on the aforementioned indicators as well as the typical values used for the resizing coefficient τ. Note that when the approximations are 'Reasonable' or 'Good', the location of the sub-optimal point is 'External' and the movement history is 'Curved', the size of the search subregion in the next iteration will not be changed, i.e., $\tau = 1$ as depicted in Table 3.1. Moreover, for 'Good' approximations, the metamodel will be reused and only the starting point will be updated in the next iteration.

3.5 Reconstitution of MAM in Python – IATRO

3.5.1 Reasons of the reconstitution of MAM

MAM was developed in Fortran programming environment and has gained improvements by different researchers over 20 years. The first version of MAM developed by Toropov [135] was written in Fortran 77 and some other developers applied Fortran 90 as the programming format. Because of the mixed programming styles, it is difficult to test and develop the framework. Moreover, although the latest Fortran standards (2003, 2008, 2015) allow programmers to write highly efficient codes with minimal efforts, it is not compatible with the older versions used in MAM. For the purpose of further investigating the characteristics of MAM, it is necessary to rewrite MAM in modern programming language.

3.5.2 New development language – Python

We choose Python as the new development language of MAM, which was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands [138]. Stack Overflow refers to Python as the 'fastest-growing major programming language' and it is reported in 2019 Stack Overflow Developer Survey that Python has edged out Java and is the second most loved language (behind Rust) [139]. Besides, Python is ranked third in the TIOBE Index [140], a measure of popularity of programming languages. It has been widely used in diverse application areas, especially in scientific and computational applications, image processing and graphic design applications, business and enterprise applications, game development and so on. It is a combination of technical features that makes Python superior to other programming languages. Some of the benefits of Python include:

1. Simple and easy to learn
   Python is a high-level programming language that coding in Python is like writing simple strict English sentences. It is easy to read and understand as the syntax of Python is almost identical to the simplified 'pseudo-code', which allows programmers to concentrate on the solution to the problem rather than the language itself.

2. Portable and extensible
   Python is an open-source language and is supported by multiple

platforms such as Windows, Linux and Mac OS. And Python is extremely extensible because it supports cross-language operations. Codes written in Java, C, or Fortran can be directly invoked via certain Python programme.

3. Extensive Support Libraries
   Python's standard library [141] provides a wide range of facilities including text processing services, string operations, web service tools, operating system interfaces, etc. In addition, various third-party libraries can be accessed from the Python Package Index [142], where most of the highly used programming tasks are already well modularized in Python libraries to enhance the productivity of developers.

3.5.3 Development environment

Following the main strategies and ideas of MAM, IATRO is developed in Python 3.7 on Ubuntu 18.04. The developed environment includes Numpy 1.17.3 [143, 144]; scikit-learn 0.22.1 [145]; Scipy 1.4.1 [146]; Numba 0.46.0 [147]; Spyder 4.0.1 [148] and AMD Ryzen 7 1800x eight-core CPU.

3.5.4 New modifications in IATRO

Besides of realizing all the functionalities of MAM, IATRO has the following improvements and modifications:

1. Modularization
   Each step in MAM as depicted in Fig. 3.1 has been modularized into an identifiable, distinct component for readability, maintainability and extensibility. Moreover, the complete algorithm of IATRO has also been modularized, i.e. it can be invoked by other functions or programmes, which makes it easier for benchmark testing.

2. Weighted least squares regression
   In MAM, the weighted least squares problems as shown in Equation 3.3.1 and Equation 3.3.7 can not be well solved when the optimization problem is large-scale or when the fitting points are very close to each other. This is mainly because of the used Fortran programmed algorithm has the efficiency problem in tackling sparse matrix equations. However, IATRO applies the 'linear regression' function provided by scikit-learn

module [145], which improves the regression performance in terms of robustness and efficiency.

3. Sequential quadratic programming (SQP) solver
   In MAM, the SQP solver is provided by the Harwell Subroutine Library [149] which has certain bugs and seems a bit outdated as 20 years have passed. Instead, the 'SLSQP' method provided by the Scipy module [146] has been in long-term maintenance and developers have fixed several critical bugs and deficiencies reported by active users. The implementation of this SQP solver in IATRO definitely improves the stability and the efficiency of the optimization process.

4. Progress curve and debug report
   In order to demonstrate the optimization process intuitively, a progress graph that shows the objective value of each iterates will be plotted once the optimization process of IATRO terminates. In addition, an adaptive debug mode is embedded in IATRO with a corresponding neat debug report. It will help the developers to locate the bugs and decrease time in developing new features.

5. Clear definition of the output solution
   In MAM, the final solution in the last iteration will be regarded as the optimal solution of the optimization problem. However, it is not guaranteed to be the best point during the optimization process. If the optimization process progresses in the wrong direction, the final solution will definitely be infeasible and should not be considered the best point found by MAM. Therefore, the best solution found by the algorithm is redefined in IATRO. By traversing the set of all evaluated points, the best solution is determined by comparing any two points according to the following rules [95]:
   (i) Any feasible solution is preferred to any infeasible solution;
   (ii) Among feasible solutions, the one having better objective function value is preferred;
   (iii) Among infeasible solutions, the one having better fitness value with smaller constraint violation is preferred.

6. Random starting point generator
   In MAM, the starting point is usually a manually specified feasible point for avoiding the risk of the optimization process getting trapped into infeasible regions in the initial stage. Although it is quite common in engineering that there exists an initial design, it is actually not the case for some highly-constrained optimization problems. Moreover, it is tedious for benchmark testing to set different starting points manually.

Thus, a random starting point generator is implemented in IATRO using the random method from the Numpy module [143, 144].

7. Benchmark functionality
   In order to reduce computational time and effort in testing the optimization performance of IATRO, the benchmark functionality has been implemented in IATRO which is formerly unsupported by MAM. This benchmark test feature allows IATRO to execute the optimization task a given number of times with user-specified parameters and retrieve the optimization results automatically. Then, a statistic report illustrating the quality of the obtained optimal solution, the termination state of each run, the performance criteria etc. will be generated accordingly.

8. Benchmark library
   For the purpose of extensively and intensively investigating the capabilities and the limitations of IATRO for solving constrained black-box optimization problems, a benchmark library which includes various well-known optimization problems has been established. The details of the problems are shown in Appendix A.

3.6 Numerical results

In this section, the performance of IATRO for solving constrained optimization problems are studied through benchmark testing. The obtained statistical results are compared with the results in literature, which clearly demonstrate the capabilities and limitations of IATRO.

3.6.1 Benchmark examples

26 well-known benchmark optimization problems are studied for testing the optimization performance of IATRO in solving CBO problems. 22 of them are chosen from the CEC'2006 test suite [93]. Note that the problem G20 and G22 in the test suit are not considered here because they are so heavily constrained that no feasible solutions have been found by any algorithms using a constraint tolerance of $1e-6$. The rest four problems are engineering design optimization problems. The welded beam design (WBD) [102] and the tension/compression spring design (TSD) [150] optimization problems are constrained problems with continuous variables. But the pressure vessel design (PVD) [150] and the speed reducer design (SRD) [104] optimization

Table 3.2: Summary description of benchmark problems

Prob.*	Optimum	d[a]	m[b]	Type[c]	ρ (%)[d]
G01	-15.0000	13	9	L+Q	0.0111
G02	-0.8036	20	2	L+N	99.9971
G03	-1.0050	10	2	N	0.0000
G04	-30665.5387	5	6	Q+N	52.1230
G05	5126.4967	4	8	L+C+N	0.0000
G06	-6961.8139	2	2	C+N	0.0066
G07	24.3062	10	8	L+Q+N	0.0003
G08	-0.0958	2	2	N	0.8560
G09	680.6300	7	4	N	0.5121
G10	7049.2480	8	6	L+N	0.0010
G11	0.7500	2	2	Q+N	0.0000
G12	-1.0000	3	1	Q+N	4.7713
G13	0.0540	5	6	N	0.0000
G14	-47.7611	10	6	L+N	0.0000
G15	961.7152	3	4	Q+N	0.0204
G16	-1.9052	5	38	L+N	0.0204
G17	8876.9807	6	8	N	0.0000
G18	-0.8660	9	13	Q+N	0.0000
G19	32.6556	15	5	N	33.4761
G21	193.7869	7	11	L+N	0.0000
G23	-400.0000	9	10	L+N	0.0000
G24	-5.5080	2	2	L+N	79.6556
WBD	1.7249	4	6	L+N	2.7020
TSD	0.0127	3	4	L+N	0.7428
PVD	6059.7143	4	4	L+N+M	39.8007
SRD	2994.4711	7	11	N+M	0.0955
MOPTA08	222.2324	124	68	N	0.0000

* Problem in bold face means that the optimum satisfies a more stringent constraint tolerance ($1e{-}6$) than the proposed value ($1e-4$) in [93]
[a] The number of design variables
[b] The number of inequality constraints
[c] The problem contains linear (L), quadratic (Q), cubic (C), nonlinear (N) functions or mixed variables (M)
[d] The ratio between the feasible region and the entire search space

problems are mixed continuous/discrete variable optimization problems. Table 3.2 shows the main properties of the benchmark problems. Here, the number of inequality constraints is different from the values listed in [93] because in TOSRBF one equality constraints $h(\boldsymbol{x}) = 0$ are converted into two inequality constraints: $g(\boldsymbol{x}) = h(\boldsymbol{x}) - \epsilon \leq 0$ and $g(\boldsymbol{x}) = -h(\boldsymbol{x}) - \epsilon \leq 0$. Note that the constraints tolerance used in IATRO is $1e-6$ which is more stringent than lots of studies. For this reason, the global optima of problems (G03, G05, G11, G13, G14, G15, G17, G21, G23) are different from the proposed values in [93]. Instead, the optimal solutions of these cases found by Aguirre [106] and Okamoto [151] are used as the known global optima.

3.6.2 Parameter settings

In order to eliminate stochastic discrepancy, 25 independent runs for each benchmark example are carried out with fixed user parameters as listed in

Table 3.3. The value of each parameter is empirically determined after numerous numerical experiments and is reasonable choice for solving general CBO problems.

3.6.3 Performance criteria

To illustrate the optimization results obtained by IATRO intuitively, various performance criteria are used as follows:

- $\boldsymbol{x}_{opt}, f(\boldsymbol{x}_{opt})$: $\boldsymbol{x}_{opt}$ is the best point encountered in the optimization process and $f(\boldsymbol{x}_{opt})$ is the corresponding objective value.
- Best, worst, average, median: The best, worst, average and median value obtained as a result from all trials.
- Feasible run and feasible rate (FR): If $\boldsymbol{x}_{opt}$ is feasible, this run is a feasible run, otherwise it is an infeasible run. And the feasible rate (FR) is equal to the number of feasible runs over total runs.
- Successful run and success rate (SR): Let $\boldsymbol{x}^*$ be the global known optimum, a run is a successful run if $f(\boldsymbol{x}_{opt}) - f(\boldsymbol{x}^*) \leq 1e-4$. And the success rate (SR) is the number of successful runs over total runs.
- ANFEs: It is used to record the average of the number of function evaluations (NFEs) when an algorithm terminates in a feasible run. If the algorithm fails to find a feasible solution in one run, the NFEs of this trial is not considered in the AENFEs.
- ENFEs: It describes the capability of an algorithm to seek a solution satisfying the success condition ($f(\boldsymbol{x}_{opt}) - f(\boldsymbol{x}^*) \leq 1e-4$) and defines as ENFEs = $^{\text{ANFEs}}/_{SR \cdot FR}$ where ANFEs is the average number of function evaluations.

Table 3.3: User parameters of IATRO with their descriptions and default values

Parameter	Description	Value
$\boldsymbol{x}^0$	The initial trial solution	Random
K_{max}	The maximum number of iterations	100
N_{plan}	The number of required sampling points	$d+5$
Δ^0	The relative size of the initial trust region	1.0
Δ_{min}	The minimum relative size of the trust region	$1e-5$
Δext	The relative size of the extended box	1.4
TOL_{con}	The constraint tolerance	$1e-6$

- AREs: It is the average number of function evaluations required by an algorithm to find the best solution in a feasible run. If the algorithm fails to find a feasible solution in one run, the NFEs of this trial is not considered in the AREs.

- EAREs: It describes the capability of an algorithm to seek the best solution and defines as EAREs = $^{\text{AREs}}/_{FR}$.

- Termination efficiency (TE): It shows the efficiency of an algorithm to terminate when there will be no possible improvements on the quality of the solution. It is defined as TE = $^{\text{AREs}}/_{ANFEs}$. If an algorithm can terminate quickly after the best solution has been found, the TR should be close to 1.0.

- Convergence graph: The graph shows the median error between the objective value of the sub-optimal solution in k_{th} iteration $f(\boldsymbol{x}^k)$ and the global known optimum $f(\boldsymbol{x}^*)$ versus the median NFEs at k_{th} iteration in all trials. The error bars mark the 25% and 75% quartile. The red point means that in median trials, the algorithm can not find a feasible sub-optimal solution at the median NFEs. And if an algorithm can obtain a feasible solution at this iteration in median runs, the point is marked as a yellow triangle. In this case, if the solution also satisfies the success condition ($f(\boldsymbol{x}^k) - f(\boldsymbol{x}^*) \leq 1e-4$), it will be denoted as a green star instead.

3.6.4 General performance

Table 3.4 shows the statistical results of the optimal objective function values obtained by IATRO on 26 benchmark problems and Table 3.5 shows the corresponding convergence statistics including the ANFEs, AREs, TE, FR, SR, ENFEs and EAREs. The convergence graph of each problem is given in Appendix B.1.

As can be seen from Table 3.4 and Table 3.5, the optimization capability of IATRO is highly dependent on the characteristics of the given optimization problem. The optimization results vary a lot from one problem to another. Among all the benchmark problems, the most difficult case to IATRO is G17 because there is no feasible solutions found by IATRO through 25 independent runs. But in 22 out of 26 problems, the FR values are all over than 96% which means IATRO shows stability in seeking feasible solutions for these cases. Only in three cases (G06, G10 and G21), the FR values are lower than 96% but it is still acceptable as the FR values are no less than 70%. However, it is

a challenging task for IATRO to find the global optima of these problems as can be observed from the SR values. Only six problems (G03, G04, G07, G16, WBD, SRD) can be solved successfully by IATRO with a high SR ($> 84\%$). In contrast, IATRO fails to find the target optima for eight problems (G02, G06, G10, G13, G17, G19, G21 and PVD), in which the SR values are 0. Besides, IATRO also has difficulty in solving another eight problems (G01, G05, G08, G11, G12, G15, G23 and SPD). The SR values of these problems range from 4% to 24%. In summary, IATRO is able to find a feasible solution of the optimization problem but is weak in achieving the global optimum.

From the convergence graphs shown in Appendix B.1, it can be verified that the global search ability of IATRO needs to be enhanced. In the majority of the problems, the optimization process terminates at either an infeasible solution or a near-optimal solution. In the meanwhile, the main characteristic of the trust-region based optimization framework like IATRO can be observed from these convergence graphs. Generally, the optimization process of IATRO can be classified into three stages. In the first stage, IATRO aims to find out the steepest descent direction where the objective function value can be optimized. As can be seen from the typical convergence graphs of the problems such as G17, G19, WBD, etc., the error between the objective function value of the current iterate and the known optimum drops significantly in the first one third of the optimization process. In the next one third of the optimization process, the median error would show a trend of fluctuating downward which is quite clear in problems such as G08, G09, G11, G13, G17, G18, G21, G23, G24, TSD and PVD. The differences between the 25% and 75% quartiles are usually large in this stage as can be seen from the error bars. In the end of the optimization process, the objective function values of the iterates tend towards stability. However, the feasibility of the final optimal solution is not guaranteed. In about one fifth of the benchmark problems (G10, G17, G19, TSD, PVD), the final solutions obtained by IATRO are not feasible in the median run.

To summarize, the optimization capability of IATRO is dependent on the types and characteristics of the targeted optimization problem. IATRO is able to obtain the global optima of some problems with high stability and efficiency but is poor in tackling other kinds of problems. In each run of the specific benchmark example, the optimization process can vary a lot. However, if we evaluate the optimization performance of IATRO from another perspective, IATRO is a practical and efficient method.

Except the eight problems (G02, G06, G10, G13, G17, G19, G21 and PVD) in which IATRO fails to find the global optima, IATRO succeeds in obtaining

Table 3.4: Statistical results of the optimal objective function values obtained by IATRO on 26 benchmark problems

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-11.4844	-13.0231	-12.6563	1.0888e+00
G02	-0.8036	-0.3812	-0.2343	-0.2939	-0.2960	4.0196e-02
G03	-1.0005	-1.0005	-0.0000	-0.9092	-1.0005	2.7098e-01
G04	-30665.5387	-30665.5396	-30665.5351	-30665.5387	-30665.5388	7.6897e-04
G05	5126.4981	5126.4981	6008.4632	5260.8809	5132.2653	2.3964e+02
G06	-6961.8139	-6961.8128	-3279.8616	-6277.3390	-6935.8967	1.2220e+03
G07	24.3062	24.3062	74.4022	26.3101	24.3062	9.8168e+00
G08	-0.0958	-0.0958	-0.0258	-0.0633	-0.0671	2.9590e-02
G09	680.6301	680.6301	1252872.5244	52888.3934	680.7164	2.5021e+05
G10	7049.2480	7049.2515	7603.9885	7129.8240	7056.6983	1.4574e+02
G11	0.7500	0.7500	0.9069	0.7737	0.7551	3.9498e-02
G12	-1.0000	-1.0000	-0.8829	-0.9514	-0.9516	3.4074e-02
G13	0.0539	0.0541	1.4532	0.7731	0.9076	3.7290e-01
G14	-47.7611	-47.7611	-47.7577	-47.7606	-47.7609	7.1784e-04
G15	961.7152	961.7152	972.1536	963.4870	962.2664	2.5307e+00
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	4.6997e-08
G17	8876.9807	-	-	-	-	-
G18	-0.8660	-0.8660	-0.5000	-0.7417	-0.8633	1.4789e-01
G19	32.6556	36.2503	214.5681	73.0934	55.0921	4.4007e+01
G21	193.7869	258.6453	723.5190	331.4278	298.2287	9.6910e+01
G23	-400.0000	-400.0000	12.1157	-316.0318	-354.3011	1.1553e+02
G24	-5.5080	-5.5080	-4.0537	-4.9765	-5.1734	5.6702e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.6380e-05
SPD	0.0127	0.0127	0.0169	0.0144	0.0142	1.3275e-03
PVD	6059.7143	6059.8611	7367.6123	6344.3215	6251.7994	2.6040e+02
SRD	2994.4710	2994.4698	2994.4711	2994.4705	2994.4707	4.5645e-04

Table 3.5: Convergence statistics of IATRO on 26 benchmark problems

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	216	138	0.64	100	12	1802	138
G02	1240	1237	1.00	100	0	-	1237
G03	771	705	0.91	100	84	918	705
G04	119	78	0.66	100	96	124	78
G05	349	337	0.96	96	8	4549	351
G06	314	211	0.67	72	0	-	294
G07	670	653	0.98	100	96	698	653
G08	233	148	0.63	100	24	973	148
G09	597	583	0.98	96	32	1943	607
G10	1171	649	0.55	72	0	-	902
G11	131	114	0.86	100	20	657	114
G12	250	154	0.62	100	4	6241	154
G13	249	235	0.94	96	0	-	245
G14	603	601	1.00	100	32	1886	601
G15	237	219	0.92	96	4	6181	228
G16	229	202	0.88	100	100	229	202
G17	-	-	-	-	-	-	-
G18	810	750	0.93	100	48	1687	750
G19	2001	1910	0.95	100	0	-	1910
G21	438	432	0.99	84	0	-	514
G23	1215	1186	0.98	100	8	15188	1186
G24	114	76	0.67	100	48	237	76
WBD	412	408	0.99	100	100	412	408
SPD	801	768	0.96	100	20	4005	768
PVD	611	243	0.40	100	0	-	243
SRD	707	556	0.79	100	100	707	556

the global optima of the rest problems definitely within certain ENFEs. Take the problem G11 as an example, the ANFEs of a feasible run, the FR value and the SR value is 131, 100% and 20% respectively. It means IATRO has a twenty percent chance of solving this problem successfully, i.e., obtaining the global optimum of this problem within 131 function evaluations. On the other hand, the corresponding ENFEs is 657, which indicates that IATRO just needs 657 function evaluations to find the global optimum of this problem at one hundred percent. Therefore, from this point of view, IATRO is able to solve 18 out of 26 problems successfully within calculated ENFEs values.

By investigating the ANFEs, AREs and TE, several problems such as G01, G04, G06, G08, G10, G12, G24 and PVD deserve more attention. These problems have a low TE (< 0.7), which means about 30% of the function evaluations are wasted in the final stage. This situation is much severe in problems G10 and PVD where more than half of the function evaluations are regarded as useless. Definitely, the termination quality of IATRO on these problems should be improved by either allowing IATRO to abort the optimization process earlier or keep on searching the optimum in a larger trust region.

3.6.5 Comparisons with state-of-the-art metaheuristic algorithms

In this section, the results of the studied optimization problems from IATRO are compared with the results of a number of state-of-the-art metaheuristic algorithms which do not involve metamodels. These algorithms are rank-iMDDE [152], LCA [113], COPSO [106], NSES [87] and Q-COM [151]. rank-iMDDE [152] is an improved constrained differential evolution (DE) method; LCA [113] is a novel algorithm which tries to model a league championship environment wherein artificial teams play in an artificial league in several weeks (iterations); COPSO [106] is a variant of particle swarm optimization method with the ability to solve constrained optimization problems; NSES [87] is an evolutionary algorithm with a self-adaptive selection method that applies multi-objective problem techniques. These algorithms are chosen because of their competitive performance in their respective metaheuristic fields. It is worth noting that making a comparison between IATRO and these algorithms in strict accordance with the relevant performance criteria is not practical because the constraint tolerance and the termination criteria of the optimization process are defined differently in these algorithms. Specifically, IATRO,

COPSO and Q-COM are able to obtain a solution with the equality constraints values not exceeding the constraint tolerance of $1e-6$ while rand-iMDDE, LCA and NSES applied a less stringent tolerance of $1e-4$ on equality constraints. As a result, the best optima obtained by these algorithms are slightly different for optimization problems with equality constraints which are marked in bold face in Table 3.2. Therefore, the comparisons in terms of the obtained best objective function values of benchmark problems are only made among IATRO, COPSO and Q-COM. Moreover, in the aforementioned methods except IATRO, a run was terminated if a solution having an objective value within $1e-4$ from the known objective value is found. This requires one to know the best-known solution of the given optimization problem, which is impractical to sophisticated real-world applications. Thus, the results reported in their literature can not reflect the optimization capability of solving a true optimization problem without the knowledge of the best-known solution.

Nevertheless, the results of IATRO (in which a run is terminated either the maximum iteration number is achieved or the search subregion is sufficiently small) are compared with the results obtained by other approaches reported in literature as shown in Table 3.6. Note that only the results of eighteen problems that can be solved successfully by IATRO are compared by other methods here and the results reported in literature are rounded to four decimal places. And the ENFEs values are calculated accordingly following the definition in Section 3.6.3. For the majority of these examples, all methods could find the global optima or near-global optima as the best objective function values listed in Table 3.6 are very similar especially for problems only including inequality constraints. Because different constraint tolerances were used, there are minor differences in the optimal objective function values of problems with equality constraints obtained by these optimizers. Compared with COPSO and Q-COM which used the same constraint tolerance of $1e-6$, IATRO can get comparative accuracy in the optimal solutions. The only special case is the problem G23 for which COPSO can not find the known optimum and the ENFEs is marked as a '-'.

Although there are no evident differences in the solving accuracy of these methods for these problems, the ENFEs values vary a lot from each other. The smallest value of ENFEs of each problem obtained by these algorithms is marked in bold face in Table 3.6. It is distinct that IATRO is the most efficient algorithm on at least eleven problems (G01, G03, G04, G07, G09, G14, G16, G18, WBD, SPD, SRD). For another seven problems, IATRO is less 'inferior' to NSES. However, as described before, the ENFEs value of NSES can just

Table 3.6: Comparisons of IATRO, rank-iMDDE, LCA, COPSO, NSES and Q-COM on eighteen problems that can be successfully solved by IATRO

Prob.	Criteria	IATRO	rank-iMDDE [152]	LCA [113]	COPSO[106]	NSES[87]	Q-COM[151]
G01	Best	-15.0000	-15.0000	-15.0000	-15.0000	-15.0000	-15.0000
	ENFEs	**1802**	80483	N.A.	95397	31710	15520
G03	Best	-1.0005	-1.0005	-1.0005	-1.0000	-1.0005	-1.0000
	ENFEs	**918**	49572	N.A.	315123	19534	18618
G04	Best	-30665.5396	-30665.5390	-30665.5387	-30665.5387	-30665.5386	-30665.5400
	ENFEs	**124**	31649	N.A.	65087	5357	15066
G05	Best	5126.4981	5126.4970	5126.4967	5126.4981	5126.4967	5126.4981
	ENFEs	4549	33615	N.A.	315257	**1558**	15117
G07	Best	24.3062	24.3062	24.3062	24.3062	24.3062	24.3062
	ENFEs	**698**	62276	N.A.	233400	171990	15274
G08	Best	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958
	ENFEs	973	2961	N.A.	6470	**541**	12762
G09	Best	680.6301	680.6300	680.6301	680.6301	680.6301	680.6301
	ENFEs	**1943**	24849	N.A.	79570	9357	15407
G11	Best	0.7500	0.7499	0.7499	0.7500	0.7500	0.7500
	ENFEs	657	7340	N.A.	315000	**135**	15015
G12	Best	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000	-1.0000
	ENFEs	6241	3101	N.A.	6647	**212**	469
G14	Best	-47.7611	-47.7649	-47.7649	-47.7611	-47.7649	-47.7611
	ENFEs	**1886**	127553	N.A.	9807000	6093	16061
G15	Best	961.7152	961.7150	961.7150	961.7152	961.7150	961.7152
	ENFEs	6181	19067	N.A.	315100	**757**	15128
G16	Best	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052
	ENFEs	**229**	18527	N.A.	40960	8982	15364
G18	Best	-0.8660	-0.8660	-0.8660	-0.8660	-0.8666	-0.8660
	ENFEs	**1687**	60084	N.A.	185654	3353	15393
G23	Best	-400.0000	-400.0551	-400.0550	-361.8566	-400.0550	-400.0002
	ENFEs	15188	205337	N.A.	-	**9757**	15458
G24	Best	-5.5080	-5.5080	-5.5080	-5.5080	-5.5080	-5.5080
	ENFEs	237	5490	N.A.	19157	**161**	15028
WBD	Best	1.7249	1.7249	1.7249	1.7249	N.A.	N.A.
	ENFEs	**412**	15000	15000	30000	N.A.	N.A.
SPD	Best	0.0127	0.0127	0.0127	0.0127	N.A.	N.A.
	ENFEs	**4005**	10000	15000	30000	N.A.	N.A.
SRD	Best	2994.4698	2994.4711	2994.4711	N.A.	N.A.	N.A.
	ENFEs	**707**	19920	24000	N.A.	N.A.	N.A.
AENFEs/AENFEs of IATRO		$^{2691}/_{2691} = 1$	$^{43157}/_{2691} \approx 16$	$^{18000}/_{1708} \approx 11$	$^{741239}/_{2034} \approx 364$	$^{17966}/_{2888} \approx 6$	$^{14378}/_{2888} \approx 5$

N.A.: The results are not available in literature.
-: The method can not obtain the global optimum of this problem, so the ENFEs is not a value.
AENFEs: The average ENFEs.
* ENFEs value with bold face means that it is the best result among all methods.

reflect when the optimal solution of the problem is found in the optimization process. If there is no prior knowledge of the true optimum, NSES should never stop within the ENFEs and it is unknown when the optimization process terminates and how much the NFEs is. Hence, it needs further analysis to determine whether NSES is better than IATRO in solving the problems such as G05, G08, G11, G12, G15, G23 and G24. Nevertheless, the superiority of IATRO in efficiency over NSES and other optimizers is apparent if the average of the efficient number of function evaluations (AENFEs) is evaluated as shown in the last line in Table 3.6. The AENFEs of each method is calculated and is in comparison with the AENFEs of IATRO. IATRO generally spends 2691 function evaluations to obtain the global optimum for each problem while rank-iMDDE requires about 43157 function evaluations, approximately 16 times more than the evaluations required by IATRO. Similarly, IATRO just requires a small fraction of AENFEs in general, about 1/11 as compared to LCA for the three engineering design problems, 1/364 as compared to COPSO for 16 problems, 1/6 and 1/5 as compared to NSES and Q-COM respectively for 15

problems.

Overall, IATRO is able to solve constrained optimization problems at the same accuracy level as metaheuristic algorithms do but reduces the required function evaluations by $5-10$ times as compared to metaheuristic algorithms which do not employ metamodels.

3.6.6 Comparisons with state-of-the-art metamodel-based algorithms

In this section, IATRO is compared with various recently proposed advanced metamodel-based algorithms in order to evaluate its performance. These approaches include COBRA [115], eDIRECT-C [118], SADE-kNN [153], SACOBRA [116] and SCGOSR [74]. COBRA [115] makes use of the radial basis function (RBF) to build metamodels of objective and constraint functions and is the first algorithm successfully solving the well-known large-scale optimization problem MOPTA08 [119]. eDIRECT-C [118] adaptively selects metamodel type (Kriging, RBF and polynomial) in the optimization process to improve the quality of approximations. Then a pure greedy search is conducted to seek the solution. SADE-kNN [153] applied the k-nearest-neighbors (kNN) technique to make predictions of unknown points in a differential evolution framework. In kNN, the approximate response of a unknown solution is a weighted average of the responses of the k nearest solutions from a database. SACOBRA [116] is based on COBRA [115] but capable of efficiently solving constrained problems with no parameter tuning. SCGOSR [74] is a multi-start optimization algorithm that select sub-optimal points from the Kriging metamodel for updating the search. To the best of the author's knowledge, these algorithms are adequate for showing the strengths and weaknesses of present metamodel-based optimization approaches.

Similar to comparing IATRO with metaheuristic algorithms, it is also not intuitive to compare these metamodel-based algorithms directly from the results shown in Table 3.7. Here are several reasons. First, except IATRO, other methods applied a relaxed tolerance ($1e-4$) on equality constraints. It is uncertain if other methods are able to obtain a more accurate solution of problems with equality constraints at this condition. Second, the optimization statistics obtained by these algorithms are not fully detailed in their papers. For example, in [116], the author only provided the median of best feasible results and average number of function evaluations for each problem studied by SACOBRA. And in [74], the range of best objective

Table 3.7: Comparisons of IATRO, COBRA, eDIRECT-C, SADE-kNN, SACOBRA and SCGOSR[a]

Prob.	Criteria	IATRO	COBRA [115]	eDIRECT-C [118]	SADE-kNN [153]	SACOBRA [116]	SCGOSR [74]
G01	Best	-15.0000	≤ -14.85	-14.9998	-15.0000	-15.0	N.A.
	EAREs	138	> 387	147	> 3722	100	N.A.
G02	Best	-0.3812	N.A.	-0.2480	-0.7429	-0.3466	N.A.
	EAREs	1237	N.A.	> 1000	N.A.	400	N.A.
G03	Best	-1.0005	≤ -0.33	-0.9989	-0.4515	-1.0	N.A.
	EAREs	705	> 451	145	N.A.	300	N.A.
G04	Best	-30665.5396	N.A.	-30665.5385	-30665.5386	-30665.539	-31026
	EAREs	78	N.A.	65	2598	200	54
G05	Best	5126.4981	≤ 5150	5145.8149	5126.49	5126.498	N.A.
	EAREs	351	13	413	> 17810	200	N.A.
G06	Best	-6961.8128	≤ -6800	-6961.8137	-6961.8138	-6961.81	-6961.8
	EAREs	294	53	35	1235	100	79
G07	Best	24.3062	≤ 25	24.3062	24.3073	24.306	24.3149
	EAREs	653	199	152	N.A.	200	178
G08	Best	-0.0958	≤ -0.09	-0.095822	-0.09582	-0.0958	-0.0958
	EAREs	148	30	154	292	200	52
G09	Best	680.6301	≤ 1000	785.6795	680.638	680.761	826.30
	EAREs	607	> 275	> 1000	N.A.	300	116
G10	Best	7049.2515	≤ 8000	7049.2484	7049.249	7049.253	N.A.
	EAREs	902	276	105	N.A.	300	N.A.
G11	Best	0.7500	N.A.	0.7499	0.7499	0.75	N.A.
	EAREs	114	N.A.	33	> 2995	100	N.A.
G12	Best	-1.0000	N.A.	-1.0000	-1.0000	N.A.	N.A.
	EAREs	154	N.A.	52	> 386	N.A.	N.A.
G13	Best	0.0541	N.A.	0.6472	0.05394	N.A.	N.A.
	EAREs	245	N.A.	> 1000	> 43907	N.A.	N.A.
G14	Best	-47.7611	N.A.	N.A.	-47.764	N.A.	N.A.
	EAREs	601	N.A.	N.A.	> 55179	N.A.	N.A.
G15	Best	961.7152	N.A.	N.A.	961.7150	N.A.	N.A.
	EAREs	228	N.A.	N.A.	> 11431	N.A.	N.A.
G16	Best	-1.9052	≤ -1.8	N.A.	-1.9051	N.A.	N.A.
	EAREs	202	38	N.A.	4633	N.A.	N.A.
G17	Best	-	N.A.	N.A.	8853.53	N.A.	N.A.
	EAREs	-	N.A.	N.A.	> 69887	N.A.	N.A.
G18	Best	-0.8660	≤ -0.8	N.A.	-0.8654	N.A.	N.A.
	EAREs	750	> 196	N.A.	> 253743	N.A.	N.A.
G19	Best	36.2503	≤ 40	N.A.	32.6632	N.A.	N.A.
	EAREs	1910	698	N.A.	N.A.	N.A.	N.A.
G21	Best	258.6453	N.A.	N.A.	193.7546	N.A.	N.A.
	EAREs	514	N.A.	N.A.	N.A.	N.A.	N.A.
G23	Best	-400.0000	N.A.	N.A.	-400.055	N.A.	N.A.
	EAREs	1186	N.A.	N.A.	> 68852	N.A.	N.A.
G24	Best	-5.5080	≤ -5	N.A.	-5.5080	N.A.	N.A.
	EAREs	76	9	N.A.	765	N.A.	N.A.
WBD	Best	1.7249	≤ 2.5	N.A.	N.A.	N.A.	1.7249
	EAREs	408	165	N.A.	N.A.	N.A.	102
SPD	Best	0.0126720	N.A.	0.012666	N.A.	N.A.	0.01267
	EAREs	768	N.A.	292	N.A.	N.A.	76
PVD	Best	7367.6123	N.A.	N.A.	N.A.	N.A.	N.A.
	EAREs	243	N.A.	N.A.	N.A.	N.A.	N.A.
SRD	Best	2994.4711	N.A.	N.A.	N.A.	N.A.	N.A.
	EAREs	556	N.A.	N.A.	N.A.	N.A.	N.A.

[a] The best results of SACOBRA are actually the median results in its paper.

function values of optimization problems obtained by SCGOSR were reported but lack of the standard deviations and other important statistics. Hence, it is hard to make comparisons of these methods by an uniform standard. Moreover, except IATRO, the termination criterion of other methods is related to the known global optimum of the test problem. COBRA, SCGOSR were set to terminate if a feasible solution with the objective function value smaller than a manually-specified target value has been found or after certain number of function evaluations. In SADE-kNN, the success threshold was $1e-4$, i.e., if a feasible solution $\boldsymbol{x}$ of which the error between the objective function value $f(\boldsymbol{x})$ and the optimum $f(\boldsymbol{x}^*)$ is smaller than this threshold, the optimization process will be terminated. eDIRECT-C applied the relative error as a termination criterion defined by $E = |\frac{f(\boldsymbol{x})-f(\boldsymbol{x}^*)}{f(\boldsymbol{x}^*)}|$. It was set to be $1e-4$ for problems only including inequality constraints and $5e-3$ for problems with equality constraints. In

SACOBRA, the results were given according to different success thresholds. For problems G02, G09 and G10, the success threshold was 0.05 while for other problems, this value was set to 0.001, which is still lager than the value used in IATRO, SADE-kNN and eDIRECT-C. Therefore, the results reported in their papers can not reflect the true performance of solving constrained optimization problems without the knowledge of the known optima. Last but not the least, most of the metamodel-based algorithms only tested a fraction of the G-problems (G01-G13) whereas IATRO tested them all. As shown in Table 3.2, it can be seen that the last nine G-problems (G14-G24) are much complex than the first thirteen G-problems according to the number of constraints or the feasible ratio. Currently, only IATRO and SADE-kNN can be compared on the performance of solving problems from G14 to G24. From Table 3.7, it is apparent that IATRO has huge advantages over SADE-kNN in solving seven problems (except G17 and G21) with respect to the required number of function evaluations. IATRO just used several tenths of function evaluations as compared to that required in SADE-kNN. For G17 and G21, the results of SADE-kNN were not satisfactory. More than 69887 function evaluations were needed by SADE-kNN to find the optimum of G17 and the EAREs of G21 were not provided in the literature. To the best of the author's knowledge, IATRO is the first metamodel-based algorithm that could successfully solve problems such as G14, G15, G16, G18 and G23 within 2000 function evaluations. Moreover, few metamodel-based algorithms have tested the mixed continuous/discrete variable optimization problems. COBRA and eDIRECT-C only tested the continuous versions of PVD and SRD, so their results are not included here. As the performance of other optimizers on these problems is unknown, it is hard to draw a firm conclusion that IATRO is the best among these methods from current insufficient results.

Nevertheless, from the results of intensively studied problems including the first thirteen G-problems, WBD and SPD shown in Table 3.7, it is quite clear that IATRO is able to obtain a solution at a higher accuracy level than other methods. Especially for G01, G04, G09 and G13, IATRO is superior to other methods in terms of both accuracy and efficiency. The results of COBRA are generally not competitive except for G08, in which COBRA were able to seek a near-global optimum within 30 function evaluations. It should be noted that in [118], Liu claimed that eDIRECT-C was superior to COBRA according to the results on problems from G01 to G13. From the results here, eDIRECT-C did perform best on five problems (G06, G07, G10, G11 and G12). SADE-kNN seems inferior to other methods because a huge number of function evaluations were required to obtain a optimum of a problem. SACOBRA performed well

on G03 and G05 but it is hard to say SACOBRA is better than IATRO because the obtained solutions were lack of accuracy. SCGOSR only tested seven problems but were good at optimizing G08, WBD and SPD. For G02, all methods fail to obtain a global or near-global optimal solution.

In summary, it is still a question that which method is the most competitive metamodel-based algorithm for solving these fifteen optimization problems as each method has its strengths and none of them shows apparent advantages over others in solving the majority of problems. But based on the aforementioned analysis, eighteen out of twenty-six benchmark problems have been successfully solved by IATRO with high efficiency and accuracy. There is no doubt that IATRO is a competitive metamodel-based algorithm for solving constrained black-box optimization problems especially within limited budget, which provides a valuable insight into the potentials of the trust-region based searching framework.

3.7 Conclusions

In this chapter, the framework of the multipoint approximation method (MAM) [77] is described first. Main strategies including the design of experiments (DOE), the metamodel building via intrinsically linear functions (ILF) and the trust region strategy are elaborated in details. The new contribution is IATRO (intrinsically linear function assisted and trust region based optimization method), which is developed in the Python environment based on the main strategies of MAM. Because MAM is written in Fortran, it has various deficiencies and shortcomings resulted from either the outdated programming language standard or the error-prone optimization process. In IATRO, there are plenty of modification and new functionalities. For example, each optimization step has been modularized; robust solvers (the weighted least squares method and the sequential quadratic programming method) are employed; the output solution is redefined; the random starting point generator is applied and a benchmark library including optimization problems of various types (linear, nonlinear, mixed variable, large-scale) is established. By comparison with several state-of-the-art metaheuristic algorithms on 18 benchmark problems, IATRO shows considerable benefits in reducing the required function evaluations without degrading the accuracy of the solution. And by comparison with several advanced metamodel-based algorithms on 26 benchmark problems, IATRO is verified to be competitive in the quality of the obtained solution and applicability of solving highly constrained and mixed-variable problems. Nevertheless, there are eight

problems (G02, G06, G10, G13, G17, G19, G21 and PVD) which can not be successfully solved by IATRO and another eight problems (G01, G05, G08, G11, G12, G15, G23 and SPD) which are solved with a low success rate. Furthermore, more attention should be paid to several problems (G01, G04, G06, G08, G10, G12, G24 and PVD), in which the termination efficiency is not satisfactory. Therefore, new strategies and modifications should be developed to enhance the optimization ability of IATRO.

4

New strategies for the enhanced IATRO

To reinforce the search abilities of IATRO ((intrinsically linear function assisted and trust region based optimization method) in every aspect, several new strategies are developed by trial and error and are implemented in the enhanced IATRO (EIATRO), which is demonstrated in details in this chapter. At first, in Section 4.1, an economical sampling strategy (ESS) that aims to reduce the required number of function evaluations is proposed. Next, Section 4.2 demonstrates a modified trust region strategy (MTRS), which achieves a good tradeoff between exploration and exploitation. And in Section 4.3, a self-adaptive normalization strategy (SANS) is described, which is designed to alleviate the difficulties in optimization caused by the objective and constraint functions of different scales. Note that each strategy can be implemented in IATRO, resulting a new method. For example, an ESS enhanced IATRO is named as IATRO–ESS (IE). And when all the strategies are activated, the method is IEMS (IATRO–ESS–MTRS–SANS). Then, the performance of the proposed strategies are studied in Section 4.4 by testing the 26 benchmark problems. Finally, a summary is given in Section 4.5.

4.1 Economical sampling strategy (ESS)

Based on the extended box selection strategy (EBS) in IATRO (Section 3.2.2), an economical sampling strategy (ESS) shown in Algorithm 3 is developed to make full use of the previous information, resulting in a reduction of the total number of required function evaluations in an optimization process. As an intrinsically linear function (3.3.2) only has $d + 1$ (d is the number of design variables) coefficients need to be determined, the minimum number of points required to fit a metamodel is $d + 1$ as well. Therefore, if several points have

been selected by EBS in a iteration of IATRO, there is no need to generate $d+1$ new points at all in this iteration.

Let n_{ext} is the number of points existing in the extended box and N_{PLAN} is the number of required sampling points in each iteration (the default is $d+5$ as shown in Table 3.3), then new points should be generated only if $n_{ext} < N_{PLAN}$. Otherwise, the number of new points n_{new} in this iteration is controlled by

$$n_{new} = N_{PLAN} - \lfloor \eta_{eco} \cdot n_{ext} \rfloor \tag{4.1.1}$$

where η_{eco} is an economical factor (the default value is 50%) meaning that in what extent the points in the extended box should be responsible for the approximations in the current search region and the symbol $\lfloor \cdot \rfloor$ indicates that the calculated value will be rounded down.

Algorithm 3: Economical sampling strategy (ESS)

Function `ESS`$(\boldsymbol{x}^k, n_{ext}, N_{plan}, \boldsymbol{A}^k, \boldsymbol{B}^k)$**:**

Input:
- $\boldsymbol{x}^k$: Starting point in k_{th} iteration.
- n_{ext}: The number of points located in the extended box
- N_{plan}: The default number of required sampling points.
- $\boldsymbol{A}^k$: Lower bounds of the current search subregion.
- $\boldsymbol{B}^k$: Upper bounds of the current search subregion.

Output:
- n_{new}: The number of new sampling points.
- X_{new}: The set of new sampling points.

if $n_{ext} \geq N_{plan}$ **then** ▷ `Points are adequate in` Q_{ext}
  $X_{new} = \varnothing$
  Return $0, X_{new}$
else ▷ `New points have to be generated`
  $n_{new} = N_{plan} - \lfloor \eta_{eco} \cdot n_{\delta} \rfloor$
  $X_{new} =$ `MSS`$(\boldsymbol{x}^k, n_{new}, \boldsymbol{A}^k, \boldsymbol{B}^k)$ ▷ `Economical sampling`
  Return n_{new}, X_{new}

4.2 Modified trust region strategy (MTRS)

As described in Section 3.4, the strategy for moving and resizing the search subregions in IATRO is based on six indicators. Through in-depth analysis of the optimization problems which can not be tackled by IATRO, several inadequacies existing in the present move limit strategy have been discovered. For example, the second indicator aims to describe the location of the obtained sub-optimal solution $\boldsymbol{x}^{k+1}$ in the present trust region Q^k (in k_{th} iteration) but it is not evaluated in a complete and comprehensive manner. Generally, a solution vector $\boldsymbol{x} \in \mathbb{R}^d$ represents a design of d design variables and each design

variable ($\boldsymbol{x}_i$) has its global lower bound A_i and global upper bound B_i. In addition, as a solution is located in a subregion of the global design space, each design variable ($\boldsymbol{x}_i$) has its current lower bound A_i^k and current upper bound B_i^k as well. Currently, a sub-optimal solution is viewed as 'External' if there exists a design variable lying either on its current lower bound or on its current upper bound. It does not take the global boundaries into consideration so that it is not clear whether the design variable which is on the current bound is also on the global bound. Moreover, this definition of 'External' is not precise, especially for variables which are actually inside the current bounds because no adjustment of the move limits on these variables is applied. Furthermore, this kind of ambiguity also exists in the third and fourth indictor which are used to reflect the movement history. These indicators evaluate the movement of the design vector in the entire d-dimensional design space but neglect the movement of an individual design variable in a one-dimensional space. Besides, the sixth indicator, which regards a solution 'Close' to the border between the feasible and infeasible regions if the maximum constraint response value is in the range $[-0.1, 0.1]$, usually results in a too rapid reduction of the sub-domain. The main reason is that the range $[-0.1, 0.1]$ is only suitable for a minority of problems and it is almost impossible to determine a typical range to distinguish between the feasible and infeasible regions because of the complexity and diversity of constrained optimization problems. Due to the aforementioned weaknesses, the move limits can only be changed isotopically, which hinders the search region from moving in the right direction to the global optimum. Therefore, a modified trust region strategy (MTRS) is developed to overcome the aforementioned shortcomings and results in a good tradeoff between exploration and exploitation abilities. The schematic description of MTRS is given in Figure 4.1 and it is based on several indicators as well.

The first indicator is actually the fifth indicator defined in IATRO which indicates the size of the current search subregion relative to the complete design space. According to this indicator, this size can be denoted as 'Large' or 'Small'. Normally, the optimization process will be terminated if the size is 'Small'.

The second indicator is used to identify the location of the each design variable x_i^{k+1} $(i = 1, \ldots, d)$ of the sub-optimal solution $\boldsymbol{x}^{k+1}$ in current bounds $[A_i^k, B_i^k]$ and in global bounds $[A_i, B_i]$. If none of the current move limits ($[A_i^k, B_i^k]$) is active, this variable x_i^{k+1} $(i = 1, \ldots, d)$ is considered an 'internal' variable. If any of the current move limits ($[A_i^k, B_i^k]$) is active but none of the global bounds $[A_i, B_i]$ is active, the location of this variable is marked as 'external'. Otherwise if one of the global bounds $[A_i, B_i]$ is active, the location is denoted

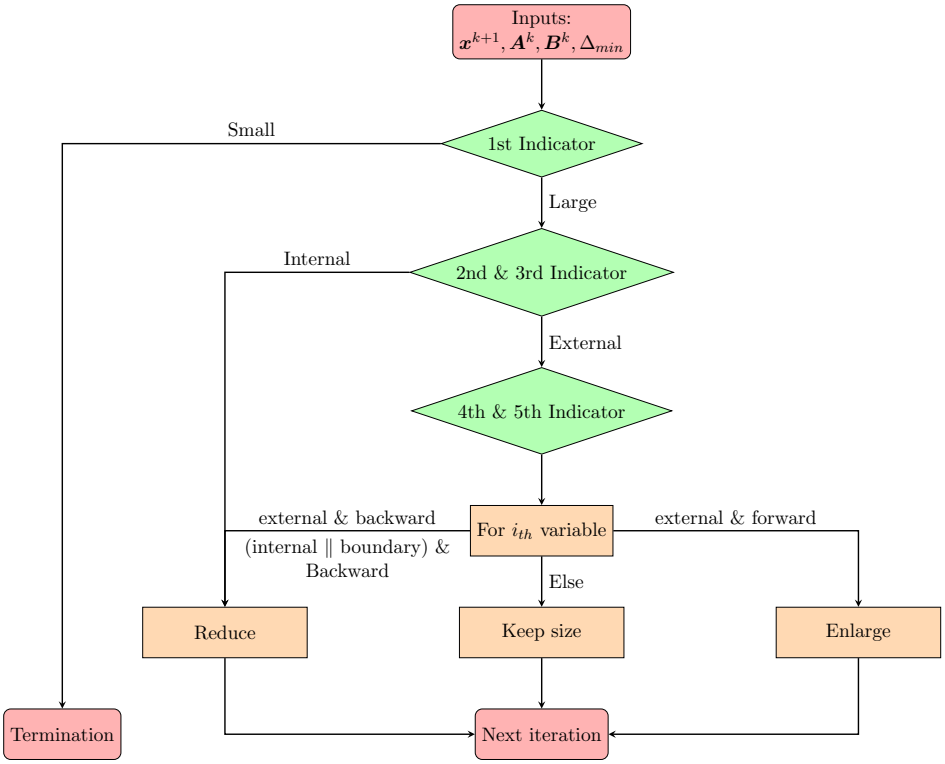


Figure 4.1: Schematic description of the modified trust region strategy

as 'boundary'.

Based on the second indicator, the third indicator indicates the location of the sub-optimal solution $\boldsymbol{x}^{k+1}$ as a whole. If none of the design variables is considered 'external', this solution is regarded as 'Internal', otherwise it is denoted as 'External'.

The fourth indicator illustrates the movement history of the each design variable $x_i^{k+1}\,(i = 1, \ldots, d)$. For simplicity, a measure $\theta_i^k\,(i = 1, \ldots, d)$ is defined as

$$\theta_i^k = (x_i^{k+1} - x_i^k) \cdot (x_i^k - x_i^{k-1})\ (i = 1, \ldots, d) \tag{4.2.1}$$

If $\theta_i^k > 0$, the movement of this variable in this dimension is considered 'forward'. Otherwise, this variable is moving 'backward'.

The fifth indicator aims to show the movement history of the sub-optimal solution $\boldsymbol{x}^{k+1}$ in the design space. Therefore, the angle between the last two move vectors is defined as

$$\varphi^k = \frac{\boldsymbol{x}^{k+1} - \boldsymbol{x}^k}{|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k|} \cdot \frac{\boldsymbol{x}^k - \boldsymbol{x}^{k-1}}{|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}|} \tag{4.2.2}$$

If $\varphi^k > 0.5$, the optimization process is moving more or less in the same

direction, so the sub-optimal solution is moving 'Forward'. If $\varphi^k <= 0$, the convergence history is marked as 'Backward'. Otherwise, the next search subregion will not be dependent on this indicator because the movement of optimal solutions is 'Uncertain'.

Based on the above indicators, the next search subregion Q^{k+1} can be determined by new move limits A_i^{k+1} and B_i^{k+1} as

$$B_i^{k+1} - A_i^{k+1} = \tau_i^k \cdot (B_i^k - A_i^k) \tag{4.2.3}$$

where τ_i^k is the resizing coefficient. In current implementation, the subregion can be resized nonisotropically. Both the enlargement and the reduction of the subspace of one design variable can be applied. When the sub-optimal solution is indeed 'Internal', a quite aggressive reduction is applied, i.e., all τ_i^k is 0.5. Otherwise, $\tau_i^k = 1.5$ is used for enlargement and $\tau_i^k = 1/1.5$ is used for reduction.

4.3 Self-adaptive normalization strategy (SANS) for objective and constraint functions

Relatively few surrogate-based algorithms focus on scaling the constraint violations to the same order of magnitude although it is a common sense that scaled constraint and objective functions can smooth the landscape of metamodel surface and then affect the efficiency and accuracy of the optimization progress. Finding a proper way of scaling is not easy. One customary approach is to carry out a pre-process before building the metamodel. This is to say, identify the min-max range of each constraint over the entire design space and then divide each constraint by the determined values [116]. Although this technique does flatten the function by avoiding the extreme value globally, it makes the optimizer even harder to escape from the flat surface because this transformation makes the function where is already somewhat flat surface much flatter. This kind of issue also occurs in the logarithmic transformation proposed by Regis [115].

Based on the aforementioned limitations, a general, easy-implemented and self-adaptive normalization strategy (SANS) is developed. In the first place, the maximum of actual responses including both the objective and constraint

values should be determined by

$$\begin{aligned} g_j^{max} &= max(|\boldsymbol{g}_j(\boldsymbol{X}^k)|) \\ f^{max} &= max(|\boldsymbol{f}(\boldsymbol{X}^k)|) \end{aligned} \tag{4.3.1}$$

where $\boldsymbol{X}^k$ is the set of fitting points which are used for metamodel building in the k_{th} iteration, f is the objective function and $g_j(\boldsymbol{x})\,(j = 1, \ldots, m)$ is the j_{th} constraint function. Secondly, the objective and constraint functions will be normalized respectively according to

$$\begin{aligned} g_j'(\boldsymbol{x}) &= \frac{g_j(\boldsymbol{x})}{g_j^{max}} \cdot \delta_g \in [-\delta_g, \delta_g], \quad if\ g_j^{max} > \delta_g \\ f'(\boldsymbol{x}) &= \frac{f(\boldsymbol{x})}{f^{max}} \cdot \delta_f \in [-\delta_f, \delta_f], \quad if\ f^{max} > \delta_f \end{aligned} \tag{4.3.2}$$

where $g_j'(\boldsymbol{x})$ and $f'(\boldsymbol{x})$ are the normalized constraint and objective functions corresponding to actual functions $g_j(\boldsymbol{x})$ and $f(\boldsymbol{x})$; δ_g and δ_f are two user-specified parameters.

Algorithm 4: Self-adaptive normalization procedure (SANS)

Function `SANS`$(\delta_f, \delta_g, \boldsymbol{f}(X^k), \boldsymbol{g}_j(X^k)\,(j = 1, \ldots, m))$**:**

Input:
- δ_f: Normalizing the objective function value into the range $[-\delta_f, \delta_f]$.
- δ_g: Normalizing the constraint function value into the range $[-\delta_g, \delta_g]$.
- $\boldsymbol{f}(X^k)$: The objective function values of the fitting points.
- $\boldsymbol{g}_j(X^k)$: The j_{th} constraint function values of the fitting points.

Output:
- $f'(\boldsymbol{x})$: The normalized objective function
- $\boldsymbol{g}_j(X^k)\,(j = 1, \ldots, m)$: The normalized constraint functions

$f^{max}, g_j^{max} = Max(|\boldsymbol{f}(X^k)|, |\boldsymbol{g}_j(X^k)|)$

if $f^{max} > \delta_f$ **then** ▷ `Objective normalization`

  $f'(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{f^{max}} \cdot \delta_f$

else

  $f'(\boldsymbol{x}) = f(\boldsymbol{x})$

if $\boldsymbol{g}_j^{max} > \delta_g$ **then** ▷ `Constraint normalization`

  $g_j'(\boldsymbol{x}) = \frac{g_j(\boldsymbol{x})}{g_j^{max}} \cdot \delta_g$

else

  $g_j'(\boldsymbol{x}) = g(\boldsymbol{x})$

Return $f'(\boldsymbol{x}), \boldsymbol{g}_j(X^k)\,(j = 1, \ldots, m)$

In this way, the constraint and objective function will be normalized adaptively during the entire optimization process. When the trust region Q^k is large, the responses of the points located in Q^k usually differ much and then they will be scaled to the same level accordingly. In this situation, SANS is similar with the usual approach. However, when the trust region Q^k is sufficiently small,

normally there will be few extreme values in objective and constraint functions in such small design space. In this situation, the conditions $g_j^{max} > \delta_g$ and $f^{max} > \delta_f$ are usually not satisfied and then there is no need to scale the functions. On the other hand, if the landscape of the function in such small region is still not flat, the SANS will be activated as normal. But since the maximum response values g_j^{max} and f^{max} generally should be smaller than the values in the initial stage, only a minor scale is sufficient enough to be applied to the original functions. Compared with the customary normalization strategy, SANS avoids the risk of normalizing the functions too much and improves the quality of approximations.

4.4 Performance study of ESS, MTRS and SANS

In order to illustrate the efficacy of the aforementioned strategies, these strategies are integrated into the framework of IATRO. For instance, by implementing the economical sampling strategy (ESS) into IATRO, the new method is named as IATRO–ESS (IE). Similarly, IATRO–ESS–MTRS (IEM) is based on IE with integration of the modified trust region strategy (MTRS). And by applying the self-adaptive normalization strategy (SANS) in IES, the method is called IEMS (IATRO–ESS–MTRS–SANS). 26 benchmark problems are used to test the optimization performance of these methods and the detailed optimization results are given in Appendix from C.1 to C.3. And the comparison results of IATRO, IE, IEM, IEMS in terms of ANFEs, FR and SR values are given in Table 4.1.

By comparing IE with IATRO, it can be observed that the ESS usually imposes little impact on the searching ability as the FR values and the SR values of both methods are very similar. But the average ANFEs value of IE is 460.41, which is only about 80% of the value obtained by IATRO. It means that ESS could reduce the computational cost in solving expensive CBO problems while hardly degrading the quality of the obtained solution. Thus, ESS is recommended to be used when the computational resource is limited.

Different from ESS, MTRS improves the global search ability a lot. By implementing MTRS into IE, the average SR value raises from 32% to 48%. This remarkable improvement is mainly because MTRS contributes significantly to the optimization process of seven problems (G08, G09, G14, G15, G19, G23 and SPD). Moreover, special attention should be given to the problem G19, for which both IATRO and IE end in failure but IEM has an 80% chance of finding the true solution. Therefore, MTRS proves to be an

Table 4.1: Comparisons between IATRO, IATRO–ESS (IE), IATRO–ESS–MTRS (IEM), IATRO–ESS–MTRS–SANS (IEMS)

Prob.	ANFEs				FR				SR			
	IATRO	IE[a]	IEM[b]	IEMS[c]	IATRO	IE[a]	IEM[b]	IEMS[c]	IATRO	IE[a]	IEM[b]	IEMS[c]
G01	216	188	297	308	100	100	100	100	12	4	20	12
G02	1240	1187	2496	2496	100	100	100	100	0	0	0	0
G03	771	799	1448	1441	100	100	100	100	84	92	96	92
G04	119	103	136	162	100	100	100	100	96	96	100	100
G05	349	200	295	189	96	96	100	100	8	0	4	0
G06	314	103	118	90	72	80	64	100	0	4	4	100
G07	670	695	890	855	100	100	100	100	96	96	100	100
G08	233	165	142	147	100	100	100	100	24	24	44	40
G09	597	592	716	744	96	92	96	100	32	48	92	100
G10	1171	827	801	804	72	40	84	92	0	0	0	12
G11	131	115	208	172	100	100	100	100	20	12	8	40
G12	250	183	225	220	100	100	100	100	4	4	4	4
G13	249	194	275	271	96	92	100	100	0	0	0	0
G14	603	618	719	792	100	100	100	100	32	12	100	100
G15	237	134	378	180	96	80	88	92	4	4	36	8
G16	229	155	174	178	100	100	100	100	100	100	100	100
G17	-	217	327	247	0.0	8	12	100	0	4	0	4
G18	810	760	932	893	100	100	100	100	48	44	52	60
G19	2001	1889	1712	1737	100	100	100	100	0	0	80	80
G21	438	256	385	391	84	100	100	100	0	0	0	0
G23	1215	884	845	897	100	96	100	100	8	12	44	44
G24	114	72	70	70	100	100	100	100	48	72	64	52
WBD	412	195	187	164	100	100	100	100	100	100	100	100
SPD	801	501	340	333	100	100	100	100	20	16	100	100
PVD	611	433	397	386	100	100	100	100	0	0	0	8
SRD	707	507	332	326	100	100	100	100	100	88	100	100
Average	579.60	460.41	570.90	557.39	92.77	91.69	94.00	99.38	32.15	32.00	48.00	52.15

[a] IATRO–ESS ($\eta_{eco} = 50\%$)
[b] IATRO–ESS–MTRS ($\eta_{eco} = 50\%$)
[c] IATRO–ESS–MTRS–SANS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)

advantageous move limit strategy which enables the optimization process to move in the right direction and converge to a satisfied solution stably.

Moreover, with the support of SANS, the optimization performance on G06, G10, G11 and G17 gets further improved as can be observed from the results of IEMS. More specifically, the other three methods can hardly solve G06 but IEMS is able to obtain the global optimum with a hundred percent. Similarly for G10, IEMS is the only method to find the optimal solution. And for G11, the average SR value of IEMS achieves 40%, which is at least two times the value obtained by other methods. Finally, G17 is a very difficult problem as all methods can not tackle it. But among them, IEMS is the most competitive method. The FR value of IEMS is 100% while other methods could scarcely generate one feasible solution. Hence, SANS proves to be a practical approach, which aims at solving non-uniform scaling problems, i.e., highly varied ranges for different constraints.

4.5 Summary

In this chapter, several strategies have been developed to improve the performance of IATRO (intrinsically linear function assisted and trust region based optimization method). As shown in Table 4.1, the implementation of economical sampling strategy (ESS), which is a new strategy, enables IATRO

to reduce 20% of the required computational cost while scarcely degrading the searching capability. The modified trust region strategy (MTRS) is a modification of the trust region strategy in MAM but it is very useful as the average SR value could increase by 50% to 48%. Moreover, the self-adaptive normalization strategy (SANS), which is also a novel approach, brings remarkable advantages in solving problems such as G06, G10, G11 and G17 whose values of constraint functions range across different orders of magnitude. Finally, IATRO–ESS–MTRS–SANS (IEMS) is capable of robustly obtaining feasible solutions of the majority of the 26 benchmark problems and finding the global optima with a satisfied success rate (52.15% in average). However, there are ten problems (G01, G02, G05, G10, G12, G13, G15, G17, G21 and PVD) that can not be stably solved and requiring further research.

5

Robust radial basis function assisted optimization framework

As stated in Section 4.5, IEMS (IATRO–ESS–MTRS–SANS) is weak in solving ten problems including G01, G02, G05, G10, G12, G13, G15, G17, G21 and PVD. But by observing the statistical results of the final solutions in Table C.5, IEMS usually obtains near-optimal solutions of these problems. This means that the optimization process of IEMS actually moves in the right direction so the main reason of the failure in obtaining the known optima lies in the inadequate approximations resulted by the intrinsically linear functions. When the original functions are complex and highly nonlinear, intrinsically linear approximations are not reliable enough in a domain of the search subregion which is located along the boundary of the feasible region. Therefore, based on IEMS, a radial basis function assisted and trust region based optimization framework (RATRO) is developed for solving constrained black-box optimization problems.

In this chapter, firstly, an overview of the RBF interpolation is given in Section 5.1. Due to the dramatic approximation performance of RBF metamodel, a balanced trust region strategy (BTRS) is developed to reach its full potentials, which is demonstrated in Section 5.2. Next, In Section 5.3, a novel design of experiments with the global intelligence is described. And for the purpose of saving as much computational budget as possible, an early termination strategy (ETS) is developed, which is discussed in Section 5.4. Then, Section 5.5 gives the comparison results to demonstrate the correctness of each strategy by testing the set of benchmark problems. Finally, a summary of the aforementioned strategies is drawn in Section 5.6.

5.1 Metamodel building using cubic radial basis function interpolation (RBFI)

As described in Section 2.3.4, given n distinct points $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathbb{R}^d$ and their corresponding function values $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)$ where $f(\boldsymbol{x})$ could be either the objective function or the constraint function, a RBF metamodel can be written as

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \cdot \phi(\parallel \boldsymbol{x} - \boldsymbol{x}_i \parallel) = \boldsymbol{w}^T \boldsymbol{\phi} \tag{5.1.1}$$

where $\parallel \boldsymbol{x} - \boldsymbol{x}_i \parallel$ is the Euclidean distance between the studied point $\boldsymbol{x}$ and the fitting point $\boldsymbol{x}_i$; ϕ represents the radial basis function. Generally, any type of the radial basis function can be used for metamodel building but the cubic form is preferred for its ease of implementation ([116, 154, 155]). Moreover, Wild et al. [156] also suggested that the quality of the approximations built by the cubic form is better than that built by other choices in Equation 2.3.36 especially under limited budget. Although the model accuracy might be improved by tuning the shape parameter γ when Gaussian ($\phi(r) = \exp(-(\frac{r}{\gamma})^2)$) or multiquadric ($\phi(r) = -\sqrt{r^2 + \gamma^2}$) form is used, the additional computational burden may be unbearable for expensive optimization problems. In current implementation, the cubic form of radial basis function with a polynomial tail is employed, i.e., the metamodel is in the form as

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \phi\left(\parallel \boldsymbol{x} - \boldsymbol{x}_i \parallel\right) + p\left(\boldsymbol{x}\right), \boldsymbol{x} \in \mathbb{R}^d \tag{5.1.2}$$

where $p(\boldsymbol{x})$ is a linear polynomial in d variables with $d+1$ coefficients as

$$p(\boldsymbol{x}) = c_0 + c_1 \cdot x_1 + c_2 \cdot x_2 + ... + c_d \cdot x_d = \boldsymbol{c^T} \cdot \boldsymbol{x} \tag{5.1.3}$$

As proposed in [116, 154], the polynomial tail $p(\boldsymbol{x})$ is beneficial to the fit of simple linear functions which otherwise have to be approximated by superimposing many RBFs in a complicated way.

To determine the coefficients $\boldsymbol{w}$ and $\boldsymbol{c}$, the following linear system of equations have to be solved:

$$\begin{bmatrix} \boldsymbol{\Phi}_{n\times n} & \boldsymbol{P}_{n\times(d+1)} \\ \boldsymbol{P}^T & \boldsymbol{0}_{(d+1)\times(d+1)} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{w}_{(n)} \\ \boldsymbol{c}_{(d+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{(n)} \\ \boldsymbol{0}_{(d+1)} \end{bmatrix} \tag{5.1.4}$$

where $\boldsymbol{\Phi}$ is an $n \times n$ square matrix containing evaluations of the RBF for the distances between all the sampling points, i.e.,

$\boldsymbol{\Phi}_{ij} = \phi(\| x_i - x_j \|)(i, j = 1, ..., n)$; $\boldsymbol{P} \in \mathbb{R}^{n\times(d+1)}$ is a matrix, of which the i-th row is $[1, \boldsymbol{x}_i^T]$; $\mathbf{0}_{(d+1)\times(d+1)} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a zero matrix; $\boldsymbol{w}_{(n)} = [w_1, \ldots, w_n]^T \in \mathbb{R}^n$; $\boldsymbol{c}_{(d+1)} = [c_0, c_1, ..., c_d]^T \in \mathbb{R}^{d+1}$; $\boldsymbol{F}_{(n)} = [f(x_1, f(x_2), ..., f(x_n)]^T \in \mathbb{R}^n$ and $\mathbf{0}_{(d+1)} \in \mathbb{R}^{d+1}$ is a vector of zeros. The coefficient matrix in Eq. 5.1.4 is invertible if it has full rank, i.e, there exits a subset of $d + 1$ linearly independent points among the fitting sets. The matrix inversion can be done efficiently by using singluar value decomposition (SVD) or similar algorithms.

5.2 Balanced trust region strategy (BTRS)

In the modified trust region strategy (MTRS) as described in Section 4.2, the size of the search subregion will be reduced by half when the sub-optimal solution is labeled as 'Internal'. It is a too rapid reduction of the search subregion which would lead to a bad convergence.

As described in Table 3.3, the proposed relative size of the initial trust region (Δ_{min}) is 1.0, i.e., the initial trust region is exactly the entire design space. In the first iteration, as the trust region is as large as the global space, the obtained sub-optimal solution will be definitely viewed as 'Internal'. So the size of the search subregion will be reduced to half of its size according to MTRS. And if this kind of reduction of the search subregion is applied several times in the first several iterations, the search subregion will be too small and is not likely to contain the global optimum. To conclude, such aggressive reduction of the search subregion limits the exploration ability of the algorithm and the optimization process might be trapped into a local optimum.

In addition, the potentials of the local exploitation can not be fully realized under such aggressive reduction in MTRS. In the final iterations of the optimization process, even if the global optimal solution is located in the search subregion, any inappropriate shrinkage of the search subregion might result in an incorrect solution. This issue becomes more severe if the problem has multiple near-optimal solutions around the near-optimal solution, for example G21. Rapid reduction of the search subregion will lead the optimization process to converge to a near-optimal solution rather than the true optimum.

Therefore, a balanced trust region strategy (BTRS) is developed to enhance both the exploration ability and the exploration ability. The schematic description of BTRS is shown in Figure 5.1. In BTRS, the third indicator

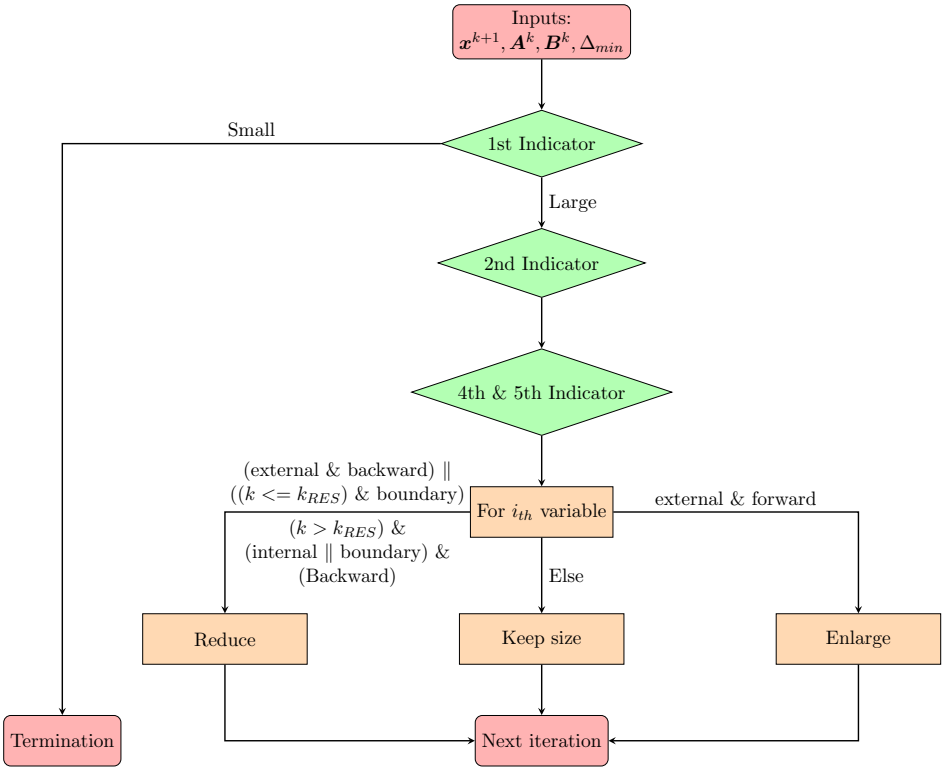


Figure 5.1: Schematic description of BTRS

which shows the location of the sub-optimal solution $\boldsymbol{x}^{k+1}$ is no longer evaluated. In this way, the solution will never be considered 'Internal' or 'External', so the search subregion will not be resized isotropically. To ensure a robust exploration of the design space, in the first k_{RES} (default value is 5) iterations, the subspace of each design variable will only be shrunk by a factor of 1.5 when this variable is located at either the global lower bound or the global upper bound, i.e., this variable is considered 'boundary'. Otherwise, the size of the subspace will remain unchanged to keep the global searching ability. And in order to achieve a moderate exploitation of the promising region where the global optimum might be located, the subspace of each design variable will only be shrunk by 1.5 when this variable is not located on the global bounds and the movement history is labelled as 'Backward'. In other conditions, the search subregion is resized following the rules defined in MTRS.

5.3 Global intelligence selection (GIS)

In IATRO and EIATRO, the DOE comprises two parts of points. The first part is the new points generated in the current search subregion by maxmin

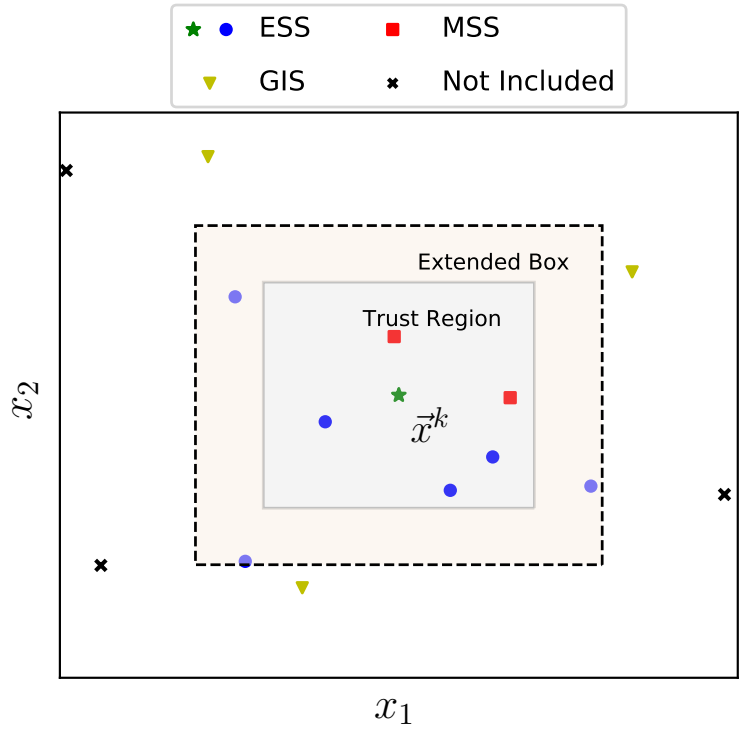


Figure 5.2: New DOE with global intelligence selection (GIS)

stochastic sampling (MSS) and the another part is the previous points that fall in the extended box, as described in Section 3.2. Because the points used to fit the approximate models are all located inside the extended box, the metamodel is adequate for the exploitation in the current trust region. And the move limit strategy such as MTRS and BTRS is responsible for the exploration in the complete design space. To further improve the overall optimization performance, the global intelligence selection (GIS) strategy is developed to improve the exploration ability while not damaging the quality of the approximations.

As shown in Algorithm 5, GIS is applied to find the points which are close to the starting point and positioned outside of the extended box. The new DOE with GIS can be described by Algorithm 6 and the points selected by GIS are marked as triangles in Figure 5.2. These points are of global intelligence for two reasons. First, as compared with the current starting point, their locations are outside of the extended box. Second, they are selected from a set of the nearest points to the current starting point. The number of points selected by GIS (n_{GIS}) is determined by the following equation:

$$n_{GIS} = \lfloor (0.5 + \frac{0.5}{k} K_{max}) \cdot N_{plan} - n_{ext} \rfloor \tag{5.3.1}$$

Algorithm 5: Global Intelligence Selection (GIS)

Function `GIS`$(\boldsymbol{x}_0, k, K_{max}, n_{ext}, Q_{ext}, N_{plan}, X_{all})$**:**

Input:

- $\boldsymbol{x}_0$: The starting point.
- k: The index of the current iteration.
- K_{max}: The maximum number of iterations.
- n_{ext}: The number of points located in the extended box.
- Q_{ext}: The extended box at k_{th} iteration.
- N_{plan}: The default number of required sampling points.
- X_{all}: The database of all sampling points.

Output:

- n_{GIS}: The number of GIS points.
- X_{GIS}: The set of points selected by GIS.

for $\boldsymbol{x}_i$ **in** X_{all} **do**
  if $\boldsymbol{x}_i \notin Q_{ext}$ **then**
    $Dist_i = \|\boldsymbol{x}_0 - \boldsymbol{x}_i\|_2$.

▷ `Define the number of required GIS points`

$n_{GIS} = \lfloor (0.5 + \frac{0.5}{k} K_{max}) \cdot N_{plan} - n_{ext} \rfloor$.

▷ `Choose` n_{GIS} `points closest to` $\boldsymbol{x}_0$ `and outside of` Q_{ext}

$X_{GIS} \subseteq \mathbb{R}^{n_{GIS} \times d}$.

Return n_{GIS}, X_{GIS}

where k is the number of the current iteration; K_{max} is the maximum number of iterations; N_{plan} is the number of required sampling points by default; n_{ext} is the number of points located in the extended box, i.e., the points selected by EBS; the symbol $\lfloor \cdot \rfloor$ means that n_{GIS} will be rounded down during the entire optimization process. From this definition, it can be seen that whether GIS is activated is dependent on the number of points positioned in the extended box (n_{ext}). If there are lots of points in the extended box, the current search subregion is usually small and the optimization process is likely to focus on exploitation, i.e., searching a vicinity of the present starting point. In this situation, GIS is not recommended to be employed because the approximations with these points are adequate enough and thus adding points outside of the extended box into fitting will ruin the approximations within the trust region. On the other hand, when there are little points falling into the extended box, the search subdomain usually has not been explored yet. In this case, the less the number of points are located in the extended box, the more the number of points are selected by GIS. Because the RBF metamodel is built by interpolation, additional global intelligence points could refine the accuracy of metamodel within the search subregion, resulting in efficiently moving the trust region along the most promising direction. From Equation 5.3.1, it is also worthwhile to note that the number of points selected by GIS will not be more than the number of required sampling points (N_{plan}). Otherwise, the quality of the approximations within the trust region can not be guaranteed.

Algorithm 6: New design of experiments (DOE)

Function `DOE`$(\boldsymbol{x}^k, k, K_{max}, \boldsymbol{A}^k, \boldsymbol{B}^k, \Delta_{ext}, N_{plan}, X_{all})$**:**

Input:

- $\boldsymbol{x}^k$: Starting point in k_{th} iteration.
- k: The index of the current iteration.
- K_{max}: The maximum number of iterations
- $\boldsymbol{A}^k$: Lower bounds of the current search subregion.
- $\boldsymbol{B}^k$: Upper bounds of the current search subregion.
- Δ_{ext}: The relative size of the extended box.
- N_{plan}: The default number of required sampling points.
- X_{all}: The database of all sampling points.

Output:

- n^k: The number of points used for fitting metamodel.
- X^k: The set of fitting points in k_{th} iteration.

if $k = 0$ **then**

▷ `In the first iteration, only maxmin stochastic sampling (MSS, Algorithm 1) is used for DOE.`

$n^k = N_{plan}$

$X^k = \texttt{MSS}(\boldsymbol{x}^k, N_{plan}, \boldsymbol{A}^k, \boldsymbol{B}^k)$

else

▷ `In next iterations, the extended box selection strategy (EBS, Algorithm 2), the global intelligence selection strategy (GIS, Algorithm 5) and the economical sampling strategy (ESS, Algorithm 3) will be used for DOE.`

$n_{ext}, Q_{ext}, X_{ext} = \texttt{EBS}(\boldsymbol{x}^k, X_{all}, \Delta_{ext}, N_{plan}, \boldsymbol{A}^k, \boldsymbol{B}^k)$

$n_{GIS}, X_{GIS} = \texttt{GIS}(\boldsymbol{x}^k, k, K_{max}, n_{ext}, Q_{ext}, N_{plan}, X_{all})$

$n_{new}, X_{new} = \texttt{ESS}(\boldsymbol{x}^k, n_{ext}, N_{plan}, \boldsymbol{A}^k, \boldsymbol{B}^k)$

$n^k = n_{ext} + n_{GIS} + n_{new}$

$X^k = X_{ext} \cup X_{GIS} \cup X_{new}$

Return n^k, X^k

5.4 Early termination strategy (ETS)

Through intensive investigation of the optimization process of EIATRO, there is a distinct drawback in termination efficiency. The only termination criterion used in EIATRO is the size indicator (the first indicator) in MTRS. As described in Section 3.4, let S^k represents the maximum ratio of the dimension length between the present trust region and the entire design space and Δ_{min} is the user-specified minimum relative size of the trust region, the optimization procedure will terminate if the current search subregion is considered 'Small', i.e., $S^k \leq \Delta_{min}$. This criterion is generally practical for the robust optimization performance when Δ_{min} is set to a very small value as $1e-5$. However, it is not economical in terms of the required number of function evaluations. Through comprehensive numerical results obtained by EIATRO, the best solution of a problem is likely to be obtained when the present search subdomain is relatively large. Therefore, for the sake of efficiency, the early termination strategy (ETS) is developed to abort

the optimization process earlier than usual if a really good solution has been found and there seems no big margin for improvement on this solution.

In the current implementation, ETS will be activated if three conditions are all satisfied. First, the obtained sub-optimal solution $\boldsymbol{x}^{k+1}$ in k_{th} iteration should be feasible because an infeasible solution is definitely not the global optimum and indicates that the optimization process should continue to run. Second, the variation of the objective function values of the last two sub-optimal solutions defined by $I = f(\boldsymbol{x}^{k+1}) - f(\boldsymbol{x}^{k})$ should be positive and less than a small value I_{max} (for example, $1e-8$). If this condition is satisfied, the optimization process possibly converges to a solution so there will be little room for reduction on the objective function value. Third, in order to avoid premature convergence, the relative size of the current search subregion S^k should be smaller than a proposed value of $\Delta_{min,2}$. From numerical tests, $\Delta_{min,2} = 0.01$ is a suitable value for solving general optimization tasks.

5.5 Comparison results

Table 5.1: Comparisons among IEMS[a], RESM[b], RESB[c], RESBG[d] and RESBGE[e]

Prob.	SR (%)					ENFEs				
	IEMS[a]	RESM[b]	RESB[c]	RESBG[d]	RESBGE[e]	IEMS[a]	RESM[b]	RESB[c]	RESBG[d]	RESBGE[e]
G01	12	24	88	96	92	2564	1538	563	445	315
G03	92	96	100	100	100	1566	1507	1453	1241	1046
G04	100	100	100	100	100	162	179	257	262	148
G05	0	100	100	100	96	-	185	94	100	57
G06	100	100	100	100	100	90	64	63	65	40
G07	100	100	100	100	100	855	602	693	620	485
G08	40	44	24	28	28	369	254	440	374	285
G09	100	100	100	100	92	744	626	667	646	605
G10	12	12	28	80	72	7279	7886	3636	778	769
G11	40	100	100	100	100	431	122	104	106	76
G12	4	40	36	40	16	5499	350	353	334	537
G13	0	24	28	40	28	-	1561	1606	1054	1287
G14	100	100	100	100	84	792	762	853	1070	902
G15	8	92	92	80	92	2442	196	173	268	120
G16	100	100	100	100	100	178	157	165	202	152
G17	4	88	52	64	60	6184	662	1144	899	924
G18	60	52	68	52	72	1488	1326	1143	1259	698
G19	80	72	48	80	88	2171	2363	3692	2023	1646
G21	0	0	48	40	40	-	-	739	952	851
G23	44	92	96	100	100	2038	285	293	294	207
G24	52	60	64	68	52	134	108	128	115	99
WBD	100	100	100	100	100	164	179	185	179	168
SPD	100	100	96	80	88	333	529	387	491	459
SRD	100	100	100	100	100	326	301	341	347	201
Average	56.17	74.83	77.83	81.87	79.17	1705.15	945.33	798.84	588.53	503.21

[a] IATRO–ESS–MTRS–SANS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)
[b] RBFI–ESS–SANS–MTRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)
[c] RBFI–ESS–SANS–BTRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5$)
[d] RBFI–ESS–SANS–BTRS–GIS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5$)
[e] RBFI–ESS–SANS–BTRS–GIS–ETS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5, I_{max} = 1e-8, \Delta_{min,2} = 0.01$)

In order to investigate the efficacy of the aforementioned strategies in improving the overall optimization performance, these strategies are implemented in the optimization framework in sequence. For example, by replacing the ILF metamodels by RBF interpolation models in IATRO–ESS–MTRS–SANS (IEMS), a new method named as RBFI–ESS–SANS–MTRS (RESM) is generated. And by replacing the modified trust region strategy (MTRS) by the balanced trust region strategy (BTRS) in RESM, the new method is called RBFI–ESS–SANS–BTRS (RESB). With integration of the global intelligence selection strategy (GIS), RESB becomes RESBG (RBFI–ESS–SANS–BTRS–GIS). And when ETS is implemented into RESBG, the new method is called RESBGE. 26 benchmark problems are tested and the detailed optimization results are given in Appendix C.3 to Appendix C.7. Since all methods can not solve G02 and PVD with a satisfied success rate, the results about G02 and PVD are not taken into consideration here to demonstrate much more clearly the optimization performance. Table 5.1 compares IEMS, RESM, RESB, RESBG and RESBGE in terms of the success rate (SR) values and the efficient number of function evaluations (ENFEs) of the 24 benchmark problems.

As can be observed from Table 5.1, it is distinct that the metamodel built by the cubic RBF with a polynomial tail is superior to that built by ILF in terms of the approximation quality. For almost all problems, RESM is more likely than IEMS to find the true optima. The average SR value of RESM is 74.83%, about 18 percentage points more than the results obtained by IEMS (56.17%). For five problems such as G05, G12, G13, G15 and G17, IEMS has an extremely low chance ($\leq 8\%$) of obtaining the optimal solutions while RESM can increase the success rate by about 10 times. And for G01, G11 and G23, RESM almost doubles the probability of finding the optima as compared to IEMS. Therefore, it can be concluded that the cubic RBF should be considered the first choice to be used to build the metamodel.

Moreover, by comparing RESM and RESB, G01 and G21 are two typical cases that can validate the superiority of BTRS over MTRS. The best, mean, worst, and median of the optimal objective function values obtained by RESM on G21 are 193.7878, 194.0552, 193.8408 and 193.8157 respectively, which are all close to the true optimum (193.7869). None of the runs can find the global optimum of G21 due to the shortage of MTRS in exploitation. But with implementation of BTRS, about half of the runs of RESB can tackle G21. In addition, G01 is a problem where RESM might be trapped into local optima due to the wrong reduction of the search subregion in the initial iterations

so the average success rate is only 24%. As BTRS enhances the exploration ability, RESB nearly quadruples this rate to 88%. Thus, BTRS proves to be more suitable than MTRS in solving CBO problems.

Furthermore, compared with RESB, RESBG shows distinct advantages in solving G10 and G19 because of the implementation of GIS. For G10, RESBG nearly trebles the SR value to 80% as compared to RESB. And the SR value of G19 obtained by RESBG is about 2 times larger than the value obtained by RESB. Meanwhile, GIS imposes positive effects on saving the computational budget. The average ENFEs value of these 24 benchmark problems obtained by RESBG is 588.53, which is only about 70% of the average ENFEs obtained by RESB. Hence, GIS could not only improve the global searching ability but also enhance the performance of convergency.

Finally, compared with RESBG, RESBGE is more efficient. The average SR value of RESBGE is about 2 percentage points lower than the value obtained by RESBG but with support of ETS, the average ENFEs value of this set of problems decreases by approximately 14.5% to 503.21. In general, ETS should be employed if the computational budget of the problem is severely limited. But if the computational resource is sufficient enough or a reduction of about 15% of the function evaluations does not make any sense, ETS is not recommended to be applied as a result of a loss of robustness.

5.6 Summary

In this chapter, four strategies (the radial basis function interpolation (RBFI), the balanced trust region strategy (BTRS), the global intelligence selection strategy (GIS) and the early termination strategy (ETS)) are developed to solve expensive constrained black-box (CBO) problems. Specifically, RBFI is an implementation of the cubic form of the radial basis function with a polynomial tail and others are first developed. Generally, RBIF is far superior to the intrinsically linear approximation, so the average SR value of RESM (RBFI–ESS–SANS–MTRS) reaches 74.83%, about 18 percentage points more than the value obtained by IEMS (IATRO–ESS–MTRS–SANS). And the BTRS addresses the deficiencies of MTRS in balancing exploration and exploitation. As a result, RESB shows obvious advantages in solving G01 where the optimum is located on the boundary of the design space and G21 where there are multiple sub-optimal solutions around the global optimum. Furthermore, the global intelligence selection (GIS) which chooses the points that are closest to the starting point

Table 5.2: User parameters in RATRO with their descriptions and default values

Parameter	Description	Value
$\boldsymbol{x}^0$	The initial trial solution.	Random
K_{max}	The maximum number of iterations.	100
N_{plan}	The number of required sampling points.	$d+5$
Δ^0	The relative size of the initial trust region.	1.0
Δ_{min}	The minimum relative size of the trust region.	$1e-5$
Δext	The relative size of the extended box.	1.4
TOL_{con}	The constraint tolerance.	$1e-6$
η_{eco}	The economical factor used in ESS indicates in what extent the points in the extended box should be responsible for the approximations in the current search subregion.	50%
$[\delta_g, \delta_f]$	The normalization coefficients used in SANS. δ_g is used for normalizing constraint functions and δ_f is used for normalizing the objective function	$[1.0, 10.0]$
k_{RES}	In the first k_{RES} iterations, the subspace of a design variable will only be shrunk when it is located on the global bounds.	5
I_{max}	The tolerated variation in objective function values used in ETS.	$1e-8$
$\Delta_{min,2}$	The tolerated relative size of the trust region used in ETS.	0.01

and outside of the current trust region into the fitting pool can not only refine the approximations in the current search subregion but also help the optimization process escape from the local optima. Therefore, RESBG (RBFI–ESS–SANS–BTRS-GIS) is able to stably solve the 24 benchmark problems (except G02 and PVD) with a success rate of 81.87% on average. Last but not the least, RESBGE (RBFI–ESS–SANS–BTRS-GIS–ETS) employs the early termination strategy (ETS) to reduce about 14.5% of the function evaluations required to obtain the global optimum as compared to RESBG. As the number of function evaluations is limited in solving expensive CBO problems, RESBGE is considered the default method.

In conclusion, the sophisticated RBF-assisted and trust region based optimization framework (RATRO) is presented in Algorithm 7 and the user parameters involved in RATRO are listed in Table 5.2. Each strategy (RBFI, ESS, SANS, BTRS, GIS or ETS) can be turned on or off for properly addressing a specific problem. When all these strategies are activated, the framework is exactly the RESBGE.

Algorithm 7: Framework of RATRO

Input:

See Table 5.2.

Output:

$\boldsymbol{x}_{opt}$: The best point encountered through the optimization process.

for k **in** $[0, K_{max}]$ **do**

Step 1. Design of Experiments (DOE) ▷ See Algorithm 6

$n^k, X^k = \texttt{DOE}(\boldsymbol{x}^k, k, K_{max}, \boldsymbol{A}^k, \boldsymbol{B}^k, \Delta_{ext}, N_{plan}, X_{all})$

Step 2. Black-box Evaluations ▷ Calculate the true responses

Objective function evaluations: $f(X^k) \subseteq \mathbb{R}^{n^k}$.

Constraint function evaluations: $g_j(X^k)\,(j = 1, \ldots, m) \subseteq \mathbb{R}^{n^k}$

Step 3. Self-adaptive Normalization ▷ See Algorithm 4

$f(\boldsymbol{x}), g_j(\boldsymbol{x})\,(j = 1, \ldots, m) =$
$\texttt{SANS}(\delta_f, \delta_g, f(X^k), g_j(X^k)\,(j = 1, \ldots, m))$

Step 4. Metamodel Building

▷ Build metamodel for the objective function and each constraint function by RBF interpolation

Approximate objective function: $\tilde{f}^k(\boldsymbol{x})$.

Approximate constraint functions: $\tilde{g}_j^k(\boldsymbol{x})\,(j = 1, \ldots, m)$.

Step 5. Solve the Approximate Subproblem

▷ Use SQP to solve the approximate optimization subproblem

$\boldsymbol{x}^{k+1} = \texttt{SQP}(\tilde{f}^k(\boldsymbol{x}), \tilde{g}_j^k(\boldsymbol{x})\,(j = 1, \ldots, m), A^k, B^k)$

Step 6. Moving Trust Region Strategy

if *Termination not satisfied* **then** ▷ ETS

▷ Update move limits by any trust region strategy, for example BTRS (Section 5.2)

$Q^{k+1} : [\boldsymbol{A}^{k+1}, \boldsymbol{B}^{k+1}]$

$k = k + 1$ ▷ Increment the iteration number

else

Traverse all the points.

Return $\boldsymbol{x}_{opt}$.

6

Strategies for large-scale optimization

In the previous chapter, RESBGE (RBFI–ESS–SANS–BTRS-GIS–ETS) is able to solve the 24 out of the 26 benchmark problems with high efficiency and robustness. The two exceptions are G02 and PVD. As stated in [90, 118, 157] and other references, G02 is a twenty-dimensional optimization problem with a periodic and multimodal objective function, which makes it impossible to be solved efficiently. Currently there are several state-of-the-art metaheuristic algorithms ([87, 90, 106]) that can obtain the best solution of G02 but the required number of function evaluations are usually of the order of magnitude of 100000, which is impractical for real-world engineering problems. Although in [151, 157] the reported function evaluations are just tens of thousands, the proposed methods have to use finite difference method to calculate the derivatives, which are usually unavailable for black-box problems. Moreover, the multimodality is too difficult to model so that there is no metamodel-based algorithm that has successfully solved this problem. Thus, it is somewhat impossible to obtain the global optimum of G02 within thousands of function evaluations. Considering that RESBGE have a 100% chance of obtaining a feasible solution of G02, in which the objective function value is usually just 50% larger than the known optimum, G02 can be considered to be partially solved by RESBGE. In addition, although the performance of RESBGE on PVD is poor, RESBGE can stably solve the another mixed continuous/discrete variable problem – SRD. Since the thesis aims at solving expensive CBO problems, how to address the mixed-variable problems is viewed as a future task.

In this chapter, the RBF-assisted and trust region based optimization framework (RATRO) is enhanced with fast computation strategy (FCS) and successive refinement strategy (SRS) and the new framework is called RATRLO (RBF-assisted and trust region based large-scale optimization

framework). It aims at tackling large-scale CBO problems but also works excellently in solving low-dimensional problems. In Section 6.1, a well-known large-scale optimization problem – MOPTA08 is introduced, along with the state of the art in solving this problem. Next, the limitations of RESBGE in solving the MOPTA08 problem are discussed. And in Section 6.2, two efficient computation approaches (the Numba compiler and parallel computation) are demonstrated in details. Then, the successive refinement strategy (SRS) for the sub-optimal solution is illustrated in Section 6.3. The following two sections (Section 6.4 and Section 6.5) show the optimization results of the new algorithm (RESBGEFS) in solving the MOPTA08 problem and the comparison results between RESBGEFS and other state-of-the-art algorithms. In addition, the improvement of SRS in solving low-dimensional benchmark problems are shown in Section 6.6. Finally, Section 6.7 summarizes the advantages of the new optimization framework and points out the remaining weaknesses in solving CBO problems.

6.1 A large-scale CBO problem – MOPTA08

The MOPTA08 benchmark problem proposed by Jones [119] is a substitute for a high-dimensional real-world problem in the automotive industry. It involves 124 decision variables and 68 inequality constraints and is considered to be a large-scale problem in the area of expensive black-box optimization as previous researches in this area mostly focused on dealing with low-dimensional problems (the number of design variables are less than 15). The objective of this problem is to determine the values of the shape variables which can minimize the mass of the vehicle subject to performance constraints (e.g., crashworthiness, harshness, durability). Each simulation of the real version of the MOPTA08 problem[1] could take 1-3 days but the simplified version provided by Jones which is based on the Kriging response surfaces to the real automotive problem takes about $0.2\,s$ in the current experimental environment (Section 3.5.3). A feasible starting point with the objective function value of 251.0706 is given by Jones [119] and the optimum appears to have the objective value of 222.74. It is highly desirable to achieve 80% of the potential reduction on the objective value to 228.0 within $1860 = 15 \cdot d(124)$ function evaluations, which corresponds to approximately one month of computational time. Moreover, it will be truly impressive if this target can be achieved with just half of the function evaluations ($992 = 8 \cdot 124$) or the objective function value can be reduced to 225 within

[1] https://www.miguelanjos.com/jones-benchmark

1860 function evaluations.

Various algorithms have been applied in [119] to solve MOPTA08, e.g., the generalized reduced gradient method (using iSIGHT-FD), the SQP (Harwell routine VF13), an evolution strategy, a local search algorithm with search directions from local surface approximations (LS-OPT), and COBYLA [92], but their performance were frustrating. The progress on the objective was either extremely slow or fast enough but no feasible solutions can be found. Regis's ConstrLMSRBF [114] was the best algorithm before 2011 compared with the Mesh Adaptive Direct Search (MADS) algorithm [158, 159] that employs a Kriging-based surrogate model, the modified LMSRBF algorithm [160], three well-known commercial-grade algorithms provided by Matlab (Fmincon, pattern search and genetic algorithm) and other approaches ([60], [161], [162]) presented at the 20th International Symposium on Mathematical Programming (ISMP 2009) conference. As stated by Regis [114], ConstrLMSRBF was the first optimization algorithm that can successfully handle a large-scale, computation-intensive black-box optimization problem. Later, Regis [115] developed COBRA, which was able to seek a feasible solution with the objective value lower than 228 within 1300 function evaluations. And recently, Bagheri [116] proposed the self-adjusting COBRA (SACOBRA), which can improve the already good mean best feasible objective value of 227.3 and 226.4 obtained by COBRA [115] and TRICEPS [163] after 1000 function evaluations respectively, to 223.3. So far, few surrogate-based algorithms have the ability to solve high-dimensional, highly-constrained and computationally expensive black-box optimization problems. Besides of the difficulty of modelling the high-dimensional responses, the computational overhead of the pure optimization process is also a big problem. As reported in [114], the average running time (excluding time spent on the function evaluations) of the ConstrLMSRBF on the MOPTA08 problem after 2000 function evaluations is about 9 hours and it increases rapidly to about 40 hours after 4000 function evaluations. This kind of computational cost of the optimization procedure is more or less unacceptable for developing and testing. Moreover, if the actual simulation time of the problem is not so much expensive, the time consumed on the optimization process will be unbearable.

In conclusion, there are three main issues in the area of metamodel-based optimization for solving large-scale, highly-constrained and computationally expensive black-box problems. First of all, besides of the massive computational overhead used by the actual simulations, the DOE process, metamodel building process and the searching process are time-consuming as

well. The total execution time of the optimization process is expected to increase exponentially with the number of design variables, which hinders the development in the area of metamodel-based optimization for solving large-scale CBO problems. Second, the metamodel usually can not provide good approximations of the true response due to the large design space. As a result, the solution of the approximate optimization problem (whether the penalty-based approach is used or not) is far away from the solution of the original optimization problem, resulting an unreliable optimization process. Last but not the least, because lots of constraints are involved, it is hard to maintain the constraint-satisfying convergence, i.e., the feasibility of solutions during the optimization process is difficult to be guaranteed.

6.2 Fast computation strategy (FCS)

6.2.1 Numba: Just in time Compiling

As described in Section 3.5.2, all proposed algorithms are developed in the Python environment for its simplicity, flexibility and maintainability. However, the performance of Python is typically much slower than C, Fortran or other comparable compiled languages because it is a dynamically typed language which requires an interpreter at runtime. Thus, if Python functions can run at a speed of Fortran functions, the issue of the massive computation time of the optimization process can be tackled. At present, different approaches have been developed to address this performance needs such as developing a just-in-time (JIT) compiler or developing a faster interpreter [164]. In the current implementation, the Numba compiler [147] which is an open-source, Numpy-aware compiler to speed up the execution time of the Python functions is applied. It uses the LLVM compiler infrastructure to JIT compile Python syntax to machine code through its collection of decorators.

An example of how to use Numba decorators to accelerate Python functions is shown in Listing 1. This function (`vec_jacobin`) is used to obtain the first order partial derivatives of the approximate functions $(f(\boldsymbol{x}), g_j(\boldsymbol{x})\,(j = 1, \ldots, m))$ at a point. The compiler directive *@njit*, which is attached to the function header, will instruct Numba to operate in *nopython* mode that forces the decorated function to run at native machine code speed. Besides, three common options can be passed to the compiler for further boosting the performance. The *cache=True* option instructs the compiler to write the compiled function into a file-based cache to avoid

```
import numpy as np
from numba import njit

@njit(cache=True,nogil=True,parallel=True)
def vec_jacobin(x,xs,d,n,eps,lamdas,cs):
        ms = lamdas.shape[0]
        jac = np.zeros((ms,d))
        jac += cs
        norm = np.zeros(n,)
        for ik in range (n):
                norm[ik] = np.linalg.norm(x-xs[ik,:])
        for k in prange (ms):
                for j in prange (d):
                        for i in prange (n):
                                jac[k,j] +=
                                ↪  lamdas[k,i]*eps**3*3*norm[i]*(x[j]-xs[i,j])
        return jac
```

Listing 1: Numba jitted function to obtain the partial derivatives of the approximate functions at a point

Table 6.1: Times and speedups of various implementations of `vec_jacobin` on MOPTA08 ($d = 124, m = 68$)

Mode	Time (s)	Speedup
Numpy	1.9457	x1
@njit	$1.1001e-3$	x1768
@njit(cache=True)	$1.0855e-3$	x1792
@njit(cache=True, nogil=True)	$1.0787e-3$	x1804
@njit(cache=True, nogil=True, parallel =True)	$1.3892e-3$	x1401

compiling the function again. The *nogil=True* option leads Numba to release Python's *global interpreter lock* (GIL) that allows the optimized machine code to run in parallel to take full advantage of multi-core machines. The *parallel=True* option enables automatic parallelizing for those operations known to have parallel semantics in that function, for example, the explicit loops and array reductions.

For illustrating the excellent performance of Numba compiler, the computational cost of `vec_jacobin` with different Numba options on the MOPTA08 problem is examined and analyzed. As summarized in Table 6.1, the performance of the basis Numpy version function is compared with the Numba version function with different options. The times recorded in Table 6.1 are the average execution time of each experiment over one hundred runs and the computed speedups are relative to the execution time of the pure Numpy implementation.

It can be seen from Table 6.1 that the pure Numpy implementation takes about 1.9457 s each time to calculate the first order partial derivatives of the approximate functions of the MOPTA08 problem at the iterate in the SQP procedure. If the approximate problem happens to have no feasible solutions, usually maximum calls (1000) of this function (`vec_jacobin`) are required for the SQP routine to terminate and output the solution. In this case, one iteration will take over 30 minutes. This kind of computation time will lead to an unbearable development cycle of new strategies for solving large-scale CBO problems. But the Numba compiler provides excellent support for efficiently operating large arrays. For example, annotating the function `vec_jacobin` with the decorator *@njit* shows an impressive speedup of $\times$1768. This result reflects the huge difference in performance that can be obtained from an interpreted code compared to a native machine code. Numba is able to support high-performance computation by amazingly reducing function running time from seconds to milliseconds. Moreover, the extra option like *cache=True* or *nogil=True* can bring a little speedup ($<$ 10%) on the pure *nopython* mode. Interestingly, using *parallel=True* option actually degrades the performance by 25% approximately, compared with the pure Numba implementation (*@njit*). One reason is that the overhead incurred by attempting to parallelize the operations in function `vec_jacobin` is significant compared to its actual running time. The input scale for this function is not big enough to show the strength of parallelism, considering the benchmark function used in [147] that coped with multiplication on 1000000 $\times$ 10 matrixes. Hence, the *parallel=True* option is not recommended to be passed to the Numba compiler for solving the MOPTA08 problem but if the studied problem has more than thousands of design variables and constraints, this Numba option should be taken into consideration. In conclusion, the function `vec_jacobin` with the decorator *@njit (cache=True,nogil=True)* shows the best performance, yielding a incomprehensible speedup of $\times$1804 compared to the pure Numpy version. Moreover, other functions conforming to the specifications of Numba can also be accelerated with the same decorator. Finally, the computational overhead of the optimization process could be decreased a lot.

6.2.2 Parallel computation

If the simulation itself does not support parallelization, then a parallel computation approach can be used for evaluating the actual objective and constraint functions. As shown in Listing 2, this function (`DOE_Basic`) distributes the input data (new generated points) across processes by using

```
import numpy as np
from multiprocessing import Pool
from itertools import repeat

def DOE_Basic(x1,d,m,nplan,ak,bk,Probname,flag_multiprocess):
    xk = Maxmin_stochastic_sampling(ak,bk,d,nplan-1,x1)
    nk = xk.shape[0]
    fk = np.zeros([nk,m+1])
    if flag_multiprocess == True:
        pool = Pool()
        fk = np.asarray(pool.starmap(Probname,zip(repeat(d,nk),
                                    repeat(m,nk), xk)))
        pool.close()
    else:
        for i in range (nk):
            fk[i,]=Probname(d,m,xk[i,:])
    return xk,fk
```

Listing 2: Parallel computation of function evaluations

the `multiprocessing` module [165] provided by the Python standard library. Then, the function evaluation process could tak full advantage of multicore CPU architectures. For the MOPTA08 problem, each function evaluation will cost about $0.2\,s$ and 129 $(d+5)$ new points usually take about $25.8\,s$ to be evaluated one by one. But with parallel execution by 8-core CPU, 129 $(d+5)$ new points only require about 5 seconds to obtain the corresponding objective and constraint values. And for real-world CBO problems, one simulation might take hours and this parallel computation technique built in the optimization programme could reduce lots of computational time.

6.2.3 Summary of the fast computation strategy

Based on RESBGE, the new method with implementation of FCS is named as RESBGEF (RBFI–EBS–SANS–BTRS–GIS–ETC–FCS). Through the performance analysis tool `pstats` [141], the execution time of subroutines involved in one iteration of RESBGE and RESBGEF in solving the MOPTA08 problem is given in Table 6.2. Generally, there are four critical procedures in terms of the computational cost: the SQP (sequential quadratic programming) procedure for solving the approximate subproblem, the function evaluations procedure to obtain the response values of the MOPTA08 problem, the DOE (design of experiments) routine to generate new points and the metamodel building process. Among them, The SQP process is the most time-consuming procedure, which is dependent on the complexity of the approximate subproblem. In the best case, only several iterations are required in SQP to output a solution of the problem but in the

Table 6.2: Time performance of one iteration in RESBGE and RESBGEF in solving the MOPTA08 problem

Description	Time (s)	
	RESBGE	RESBGEF
SQP	462.340	2.267
Function evaluations	24.030	5.024
DOE	5.313	0.112
Metamodel building process	1.500	1.499
Total	493.894	10.002

worst case, there might be no solution of the problem due to the approximation error, resulting a long time search until the maximum number of iterations (1000 by default) is achieved. And in the SQP process, the most time-consuming part is to call function `vec_jacobin` (Listing 1) to obtain the partial derivatives of the functions at the iterates. As shown in Table 6.1, one call of function `vec_jacobin` of the Numpy version takes about 2 seconds while the Numba version only takes about milliseconds. As a result, the average time of the SQP process (considering there are 300 iterations) in RESBGEF only requires about 2 seconds, approximately 1/200 of the time in RESBGE. Moreover, FCS could also reduce the runtime of the function evaluation process and the DOE process by about five to ten times. And finally, the total computation time of one iteration of the optimization process could decrease from 8 minutes to about 10 seconds by FCS. Only in this way, the framework for solving large-scale optimization problems could be further developed.

6.3 Successive refinement strategy (SRS) for the sub-optimal solution

As described in Section 6.1, the first obstacle in solving large-scale CBO problem is the high computational overhead of the optimization process. This issue has been settled by employing the fast computation strategy (FCS). Then the remaining difficulty in solving the high-dimensional, severely-constrained optimization problem like MOPTA08 is to ensure the metamodel as accurate as possible and to maintain the feasibility through the optimization process. The successive refinement strategy (SRS) is developed to tackle this issue by refining the quality of the sub-optimal solution in each iteration.

As shown in Algorithm 8, SRS is an inner iterative procedure in the global optimization process to obtain a refined solution of the approximate optimization subproblem. The iteration number in SRS is denoted as t and the global iteration number is labelled as k. Given the starting point $\boldsymbol{x}^{k,0}$ in the first iteration in SRS ($t = 0$), the solution ($\boldsymbol{x}^{k,1}$) obtained by the SQP solver can be considered an unrefined solution of the original approximate subproblem. But in the next iteration, the starting point in the previous iteration $\boldsymbol{x}^{k,t}$ will be added into the fitting points $X^{k,t}$, leading to refined metamodels. Then, the solution $\boldsymbol{x}^{k,t+1}$ of the former approximate optimization subproblem serves as the starting point for the SQP procedure to solve the new updated approximate subproblem. This procedure is repeated until certain termination criteria are satisfied. And the final solution in SRS $\boldsymbol{x}^{k,t+1}$ will be considered the solution in the k_{th} iteration of the global optimization process. In general, SRS can be seen as a simplified metamodel-based optimization framework without the moving trust region as compared to RESBGE. Thus, SRS is able to generate a better sub-optimal solution through successive refinement than the solution directly obtained through an unrefined approximate subproblem.

By setting various termination criteria, SRS can have flexible optimization performance. Five main termination criteria are illustrated in the following paragraph and the default parameter settings are summarized in Table 6.3.

First, SRS will terminate if the maximum number of iterations t_{max} is reached. It is advised to be used in any case for avoiding going into an infinite loop. By default, the value of t_{max} is $d + 1$ where d is the number of design variables because it is usually sufficient enough for SRS to obtain a refined solution through $d + 1$ iterations.

Second, SRS should abort smartly if there is no need for refinement at all. This can be judged by the variation of the objective function values of the last two iteration points in SRS, defined by

$$I_{SRS} = |f(\boldsymbol{x}^{k,t+1}) - f(\boldsymbol{x}^{k,t})| \tag{6.3.1}$$

If $I_{SRS} <= I_{max,2}$ where $I_{max,2}$ is a small positive value such as $1e - 6$, there is little room for refining the solution so the procedure has to stop earlier than normal. In order to make full use of computational resources, this termination criterion is also recommended to be used for general optimization tasks.

Third, SRS could break up earlier if the solution has been well refined, i.e., the solution obtained by the SQP solver is feasible. In this case, this solution

Algorithm 8: Successive refinement strategy (SRS)

Function `SRS(`$\boldsymbol{x}^{k,0}, k, X^{k,0}, \boldsymbol{f}(X^{k,0}), \boldsymbol{g}(X^{k,0}), \boldsymbol{A}^k, \boldsymbol{B}^k, t_{max}$`):`

Input:

- $\boldsymbol{x}^{k,0}$: The starting point in k_{th} iteration in the global optimization process, i.e., the initial point in the first iteration in SRS procedure.
- k: The index of the current iteration.
- $X^{k,0}$: The fitting points in the current iteration.
- $\boldsymbol{f}(X^{k,0}), \boldsymbol{g}(X^{k,0})$: The corresponding objective and constraint values of the fitting points.
- $\boldsymbol{A}^k, \boldsymbol{B}^k$: The lower and upper bounds of the search subregion.
- t_{max}: The maximum number of iterations in SRS. The default is $d+1$.

Output:

- $\boldsymbol{x}^{k+1}$: The refined solution of the approximate subproblem.

for t **in** $[0, t_{max}]$ **do**

▷ `Build/Update metamodels using the RBF interpolation`

$\tilde{f}^{k,t}(\boldsymbol{x}), \tilde{g}_j^{k,t}(\boldsymbol{x})\,(j=1,\ldots,m) =$ `RBF(`$X^{k,t}, \boldsymbol{f}(X^{k,t}), \boldsymbol{g}(X^{k,t})$`)`

▷ `Use SQP to solve the approximate optimization subproblem`

$\boldsymbol{x}^{k,t+1} =$ `SQP(`$\tilde{f}^{k,t}(\boldsymbol{x}), \tilde{g}_j^{k,t}(\boldsymbol{x})\,(j=1,\ldots,m), \boldsymbol{x}^{k,t}, \boldsymbol{A}^k, \boldsymbol{B}^k$`)`

if *Termination criteria satisfied* **then**

break

else

▷ `Update the pool of fitting points`

$X^{k,t+1} = X^{k,t} \cup \boldsymbol{x}^{k,t}$

$\boldsymbol{f}(X^{k,t+1}) = \boldsymbol{f}(X^{k,t}) \cup f(\boldsymbol{x}^{k,t})$

$\boldsymbol{g}(X^{k,t+1}) = \boldsymbol{g}(X^{k,t}) \cup g(\boldsymbol{x}^{k,t})$

$t = t+1$ ▷ `Increment the iteration number` t

▷ `Output the refined solution`

$\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k,t+1}$

Return $\boldsymbol{x}^{k+1}$

$\boldsymbol{x}^{k,t+1}$ can be considered a good sub-optimal solution in the k_{th} iteration. Usually, this termination criterion results in a more economical optimization process in terms of function evaluations and is suggested to be applied as well.

Fourth, if the total number of function evaluations is limited, SRS is certain to come to an end when the maximum number of function evaluations ($NFEs_{max}$) is achieved. Take the MOPTA08 problem as an example, Bagheri [116] conducted numerical experiments in solving this case within 1000 function evaluations. For making clear comparisons between Bagheri's SACOBRA and the optimization method with SRS, this limitation should be considered in the optimization process. Buf if the computational budget of an optimization problem is adequate, this termination criterion will not be triggered.

The fifth termination criterion is used to control whether the SRS should be activated. For a problem with severely limited computational budget, refining the quality of the solutions in the first several iterations is more or

Table 6.3: Five termination criteria used in the successive refinement strategy (SRS) for solving the MOPTA08 problem

Parameter	Description	Value
t_{max}	The maximum number of iterations in SRS.	$d+1$
$I_{max,2}$	If the variation of the objective function values of the last two iteration points in SRS is smaller than $I_{max,2}$, SRS will terminate.	$1e-6$
$SRS_{feasible}$	SRS will be stopped if the iteration point $\boldsymbol{x}^{k,t}$ is feasible.	True
$NFEs_{max}$	SRS will halt if the maximum number of function evaluations ($NFEs_{max}$) is achieved.	1000
η_{SRS}	SRS will be disabled in the first $\eta_{SRS} \cdot NFEs_{max}$ function evaluations	0.25

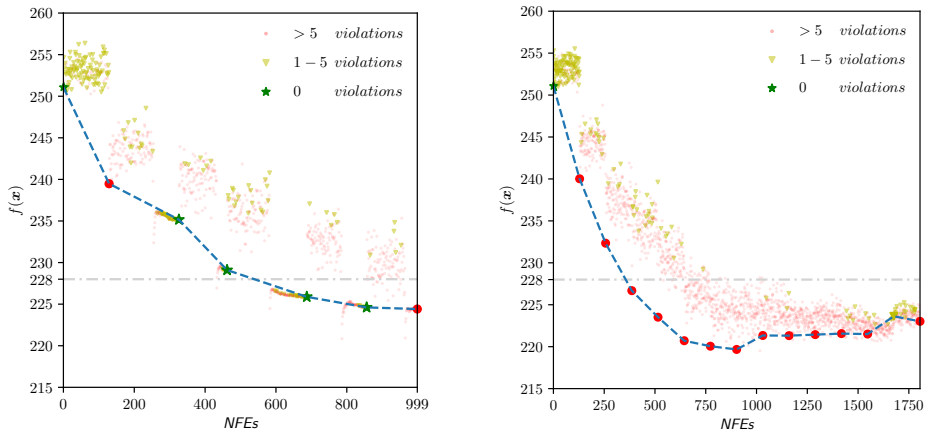
Table 6.4: Specific parameters and their values in RESBGEFS for solving the MOPTA08 problem

Parameter	Description	Value
$\boldsymbol{x}^0$	The initial point given by Jones [119].	$f(\boldsymbol{x}^0) = 251.0706$
K_{max}	The maximum number of iterations.	30
Δ^0	The relative size of the initial trust region.	0.1
TOL_{con}	The constraint tolerance.	$1e-4$
$NFEs_{max}$	The optimization process will terminate if the maximum number of function evaluations ($NFEs_{max}$) is achieved.	1000

less unworthy because they are usually far away from the true optimum. Thus, the successive refinement procedure is suggested to be employed after several iterations or after a number of function evaluations. Specifically, if the maximum number of function evaluations $NFEs_{max}$ has been set, SRS will not be activated within the first quartile of the function evaluations for better efficiency.

6.4 RESBGEFS on the MOPTA08 problem

Based on RESBGEF, the new method with the successive refinement strategy is called RESBGEFS (RBFI–EBS–SANS–BTRS–GIS–ETC–FCS–SRS). Since Jones [119] provided a feasible point of the MOPTA08 problem with the objective function value of 251.0706 and the computational budget are severely limited, an

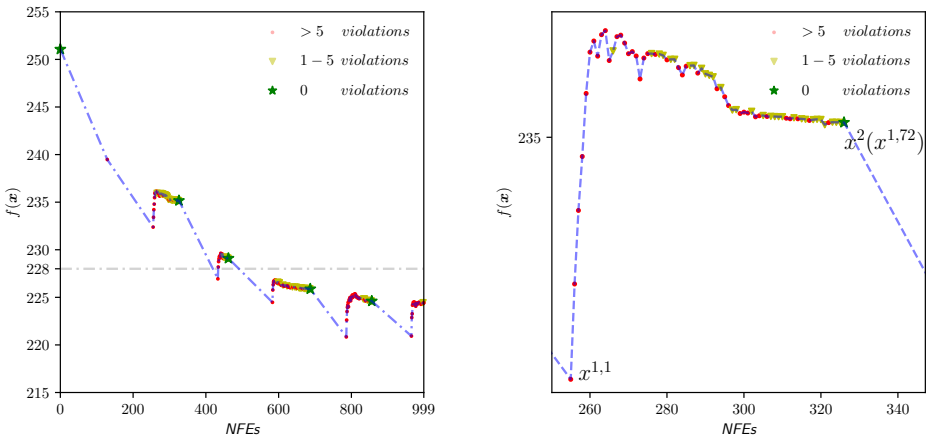


(a) RESBGEFS (b) RESBGEF

Figure 6.1: A typical progress curve of RESBGEFS and RESBGEF on the MOPTA08 problem

optimization algorithm should focus on finding a local optimum around this feasible solution rather than searching the global space. For the purpose of carrying out the local exploitation, the relative size of the initial search subregion in RESBGEFS is suggested to be set as $\Delta^0 = 0.1$, i.e., only one tenth of the entire design space will be explored in the first iteration. In addition, the optimization process is expected to be terminated after 1000 function evaluations as proposed in [119]. But because in RESBGEFS, the number of new sampling points in each iteration is flexible (See Algorithm 6), RESBGEFS might not terminate exactly at 1000 function evaluations. And for the reason that the points generated by maxmin stochastic sampling (See Algorithm 1) are usually not feasible, RESBGEFS should stop before this sampling process if the number of new required points plus the number of previous sampled points will exceed 1000. Moreover, as described in [115, 116, 119], the MOPTA08 problem is so highly constrained that finding a totally feasible solution (constraint values are all smaller than 0) is difficult, $1e-4$ would be a reasonable value of the constraint tolerance in this case. Table 6.4 summarizes the specific parameter values used for solving the MOPTA08 problem and other parameters and their values are kept the same as shown in Table 5.2 and Table 6.3.

Figure 6.1 compares the typical progress curve of RESBGEFS and RESBGEF on the MOPTA08 problem under the same parameter settings except the maximum number of function evaluations $NFEs_{max}$. Since RESBGEF can not optimize the MOPTA08 problem well, $NFEs_{max}$ is set to be 1860 to see if RESBGEF can obtain a solution after more function evaluations. The blue dashed line indicates the optimization trajectory of an algorithm, where



(a) All solutions (b) Solutions in one SRS procedure

Figure 6.2: A trajectory of solutions obtained by the SQP solver in RESBGEFS in solving the MOPTA08 problem

each point is a solution of the approximate sub-problem in the corresponding iteration. The points which are not located on this line are the sampling points in the entire optimization process and each point represents a design. A design is marked as a red point if there are more than five constraints that are violated and if it is a feasible design, it will be marked as a green star. Otherwise, it is denoted as a yellow triangle. The horizontal light grey line means an objective function value of 228, which is considered a good feasible objective function value for the algorithm to aim for. It is clear from Figure 6.1 that the MOPTA08 problem is so severely-constrained that no feasible solutions could be found by RESBGEF within 1860 function evaluations although the objective function values of iteration points drop quickly. As pointed out in [115, 119], a great deal of algorithms can not maintain feasibility through iterations and they take a long time to resolve constraints like RESBGEF. But with SRS, four feasible iteration points are found after successive refinements, which are marked as green stars. The solution in the last iteration ($NFEs = 999$) is infeasible because the refinement procedure is ended due to the fact that the maximum number of function evaluations is achieved. It is extremely impressive as about 90% of the potential reduction on the objective function value to 225 can be achieved by RESBGEFS within 1000 function evaluations.

In order to further illustrate the performance of SRS, the corresponding trajectory of points obtained by the SQP solver in the entire optimization process (Figure 6.1a) is given in Figure 6.2a and the more specific view of the solutions in SRS in the second iteration is shown in Figure 6.2b. As can be seen from Figure 6.2b, there are three stages in SRS. The initial solution $\boldsymbol{x}^{1,1}$ of the approximate subproblem in the SRS is severely infeasible (the number

of violated constraints is more than 5) with a quite small objective function value. So in the first several iterations in SRS, the obtained solution moves to the feasible region quickly with an increase of the objective function value. Later, as the updated approximations within the trust region become more and more adequate, the obtained solutions in SRS are less infeasible. There is a steady decrease of the objective function value because the solution is moving to the optimum of the approximate subproblem in the current search subregion. In the last stage of SRS, the refined solutions are very close to the optimum, leading to a stable trajectory of the corresponding objective function values. But because the MOPTA08 problem is a severely constrained and high-dimensional problem, the constructed RBF models are too difficult to become as accurate as the original functions. As a result, it still takes a number of iterations in the final stage of SRS to obtain a feasible solution of the MOPTA08 problem. In conclusion, through the successive refinement strategy, the approximations in the current search subregion become more and more accurate and the objective function value of the solution of the approximate subproblem increases at first, then decreases and finally remains nearly unchanged until a feasible solution is found.

6.5 Comparisons between RESBGEFS and other state-of-the-art algorithms in solving the MOPTA08 problem

To further verify the performance of SRS in solving the MOPTA08 problem, 10 independent runs of RESBGEFS are carried out with the same given starting point. Figure 6.3 shows the median objective value of the sub-optimal solution in k_{th} iteration versus the median NFEs at k_{th} iteration in ten trials. It can be seen from this figure that the last 4 iteration points are all feasible, which proves the robustness of SRS in improving the quality of the solution. In addition, it can be concluded that within 1000 function evaluations, RESBGEFS is able to obtain a refined feasible solution with the median objective function value of 225.0, which meets the desired target proposed by Jones [119].

Compared with other state-of-the-art algorithms such as ConstrLMSRBF [114], COBRA [115], TRB[166] and SACOBRA [116], RESBGEFS is very competitive in optimizing the MOPTA08 problem from the results summarized in Table 6.5. More specifically, RESBGEFS is just a little inferior to SACOBRA but is far superior to other algorithms in solving the

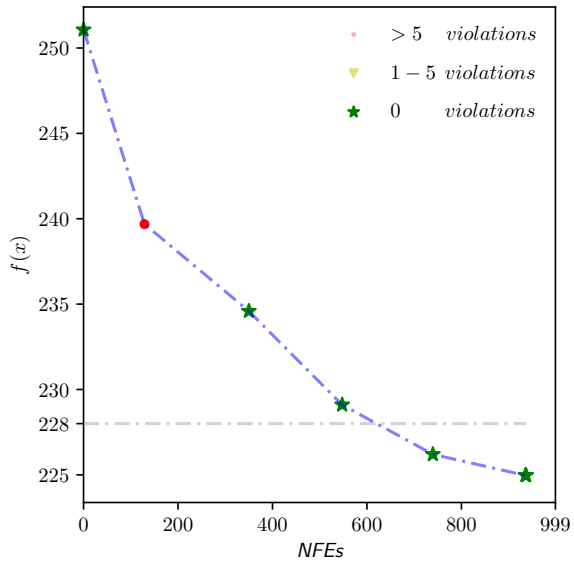


Figure 6.3: The convergence graph of RESBGEFS in solving the MOPTA08 problem: the median objective value of the sub-optimal solution in k_{th} iteration versus the median NFEs at k_{th} iteration in 10 trials.

Table 6.5: Comparing different algorithms on optimizing the MOPTA08 problem within 1000 function evaluations

Algorithm	Best	Worst	Mean	Median
ConstrLMSRBF [114]	> 228	N.A.	N.A.	N.A.
COBRA [115]	226.3	229.5	227.3	227.0
TRB[166]	225.5	227.4	226.4	226.2
SACOBRA [116]	223.0	223.8	223.3	223.3
RESBGEFS	224.5171	225.8582	224.9682	224.9683

MOPTA08 problem. SACOBRA works excellently in solving the MOPTA08 problem as it achieves 95% of the potential reduction on the objective function value to 223.0, which is five percentage points more than the result obtained by RESBGEFS. But its overall performance in solving constrained optimization problems is not as good as RESBGEFS, which will be discussed later. Besides, the average computational time of RESBGEFS on the MOPTA08 problem after 1000 function evaluations is just about 30 minutes, which is a small fraction of the time required by ConstrLMSRBF. Although the computational time of other algorithms on this problem was not reported in their papers, there is a reason to believe that RESBGEFS is more suitable for solving large-scale optimization problems due to the Numba compiler. To sum up, RESBGEFS shows competitive advantages in solving the MOPTA08 problem with a limited number of function evaluations. In other words, RESBGEFS proves to be a practical method in solving

severely-constrained and high-dimensional CBO problems even if the computational resources are limited. Specifically, the Numba compiler remarkably improves the computation speed and reduces the development time of new strategies. Furthermore, the successive refinement strategy (SRS) is the key to obtaining a feasible solution of the high-dimensional CBO problem with high efficiency and robustness. With further in-depth research on when to apply and when to terminate this procedure in the future, RESBGEFS will definitely have better performance in solving the MOPTA08 problem and other high-dimensional problems.

6.6 RESBGEFS on low-dimensional constrained benchmark problems

6.6.1 Optimization results of RESBGEFS on 26 benchmark problems

Although the successive refinement strategy (SRS) is originally developed to solve high-dimensional CBO problems like MOPTA08 to refine the sub-optimal solution, it works excellently in solving low-dimensional CBO problems as well. Table 6.6 shows the statistical results of the optimal objective function values obtained by RESBGEFS (RBFI–EBS–SANS–BTRS–GIS–ETS–FCS–SRS) on 26 low-dimensional benchmark problems and Table 6.7 gives the corresponding convergence statistics. In all the numerical experiments, SRS works in the default mode as described in Section 6.3 and the values of other user parameters are kept the same as listed in Table 5.2.

Similar to other methods proposed earlier, G02 and PVD are two most difficult problems to RESBGEFS. Although SRS does not improve the performance in optimizing PVD (a mixed-variables optimization problem), it alleviates the difficulty in solving G02 a lot. The best objective function value of G02 obtained by RESBGEFS is -0.7931, which is quite close to the known global optimum -0.8036 while this value obtained by other methods (for example, RESBGE and RESBGE) without applying SRS are larger than -0.6020. This is a strong evidence that SRS works well especially for complex problems as G02 is a multi-modal and twenty-dimensional problem. This problem needs further study but to the best of the author's knowledge, RESBGEFS is the among the best metamodel-based algorithm in optimizing this problem.

Table 6.6: Statistical results of the optimal objective function values obtained by RESBGEFS on 26 benchmark problems

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-13.8281	-14.9531	-15.0000	2.2964e-01
G02	-0.8036	-0.7931	-0.1769	-0.4597	-0.5223	2.0145e-01
G03	-1.0005	-1.0005	-1.0005	-1.0005	-1.0005	1.3482e-06
G04	-30665.5387	-30665.5394	-30665.5387	-30665.5387	-30665.5387	1.7935e-04
G05	5126.4981	5126.4981	5126.4981	5126.4981	5126.4981	2.0614e-06
G06	-6961.8139	-6961.8139	-6961.8139	-6961.8139	-6961.8139	1.2526e-05
G07	24.3062	24.3062	24.3062	24.3062	24.3062	1.4092e-06
G08	-0.0958	-0.0958	-0.0000	-0.0518	-0.0291	3.5587e-02
G09	680.6301	680.6301	680.6305	680.6301	680.6301	1.3229e-04
G10	7049.2480	7049.2480	7098.4572	7051.2164	7049.2480	9.6430e+00
G11	0.7500	0.7500	0.7500	0.7500	0.7500	2.5893e-07
G12	-1.0000	-1.0000	-0.7649	-0.9417	-0.9694	7.0086e-02
G13	0.0539	0.0539	1.0000	0.3368	0.4389	3.0424e-01
G14	-47.7611	-47.7611	-47.7603	-47.7611	-47.7611	1.6014e-04
G15	961.7152	961.7152	961.7152	961.7152	961.7152	2.6580e-07
G16	-1.9052	-1.9052	-1.9017	-1.9048	-1.9052	1.0925e-03
G17	8876.9807	8853.5401	8929.5707	8878.3353	8868.8727	2.6931e+01
G18	-0.8660	-0.8660	-0.5000	-0.7400	-0.8660	1.5349e-01
G19	32.6556	32.6556	32.6556	32.6556	32.6556	7.3112e-07
G21	193.7869	193.7861	193.7879	193.7870	193.7870	3.1476e-04
G23	-400.0000	-400.0002	-400.0000	-400.0000	-400.0000	4.2075e-05
G24	-5.5080	-5.5080	-4.0537	-4.9459	-5.5080	6.8258e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	9.8695e-12
SPD	0.0127	0.0127	0.0181	0.0133	0.0127	1.6501e-03
PVD	6059.7143	6059.7143	6431.1571	6178.2636	6131.6372	9.5270e+01
SRD	2994.4710	2994.4702	2994.4711	2994.4708	2994.4710	2.9666e-04

Table 6.7: Convergence statistics of RESBGEFS on 26 benchmark problems

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	300	209	0.70	100	96	313	209
G02	1537	1232	0.80	100	0	-	1232
G03	470	413	0.88	100	100	470	413
G04	93	28	0.31	100	100	93	28
G05	74	43	0.59	100	100	74	43
G06	44	24	0.54	100	100	44	24
G07	334	330	0.99	100	100	334	330
G08	84	60	0.72	100	24	350	60
G09	293	291	1.00	100	88	332	291
G10	223	194	0.87	100	96	232	194
G11	105	59	0.56	100	100	105	59
G12	86	58	0.68	100	24	357	58
G13	209	166	0.79	100	44	475	166
G14	740	711	0.96	100	88	841	711
G15	101	62	0.62	92	92	120	68
G16	80	41	0.51	100	88	91	41
G17	446	333	0.75	100	68	656	333
G18	188	156	0.83	100	52	361	156
G19	1139	1062	0.93	100	100	1139	1062
G21	319	250	0.78	100	48	664	250
G23	98	53	0.55	100	100	98	53
G24	56	23	0.41	100	56	100	23
WBD	120	99	0.83	100	100	120	99
SPD	256	223	0.87	100	88	291	223
PVD	382	78	0.20	100	4	9556	78
SRD	121	61	0.50	100	100	121	61

6.6.2 Comparisons between RESBGEFS and RESBGE on 24 benchmark problems

To further illustrate the advantages of SRS in solving constrained CBO problems, comparisons between RESBGEFS and RESBGE on 24 benchmark problems (except G02 and PVD) are made as shown in Table 6.8. Because these problems have already been stably solved by RESBGE, SRS can not bring extremely distinct improvement on the SR values of these problems. The average SR value of RESBGEFS is 81.33%, about 2 percentage points more than that obtained by RESBGE. But the required number of function evaluations for each problem is decreased dramatically due to the application of SRS. On average, the ENFEs of RESBGEFS is 324.19, which is just 64.42% of the ENFEs obtained by RESBGE (503.21). In conclusion, SRS could not only improve the performance in optimizing large-scale CBO problems but also reduce the required NFEs in solving low-dimensional CBO problems remarkably.

Table 6.8: Comparisons between RESBGEFS[a] and RESBGE[b]

Prob.	SR (%)		ENFEs		EAREs	
	RESBGEFS[a]	RESBGE[b]	RESBGEFS[a]	RESBGE[b]	RESBGEFS[a]	RESBGE[b]
G01	96	92	313	315	209	177
G03	100	100	470	1046	413	1012
G04	100	100	93	148	28	62
G05	100	96	74	57	43	43
G06	100	100	44	40	24	23
G07	100	100	334	485	330	475
G08	24	28	350	285	60	54
G09	88	92	332	605	291	553
G10	96	72	232	769	194	463
G11	100	100	105	76	59	51
G12	24	16	357	537	58	68
G13	44	28	475	1287	166	333
G14	88	84	841	902	711	731
G15	92	92	120	120	68	97
G16	88	100	91	152	41	121
G17	68	60	656	924	333	523
G18	52	72	361	698	156	438
G19	100	88	1139	1646	1062	1419
G21	48	40	664	851	250	285
G23	100	100	98	207	53	138
G24	56	52	100	99	23	30
WBD	100	100	120	168	99	154
SPD	88	88	291	459	223	391
SRD	100	100	121	201	61	145
Average	81.33	79.17	324.19	503.21	206.54	324.31

[a] RBFI–ESS–SANS–BTRS–GIS–ETS–SRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5, I_{max} = 1e-8, \Delta_{min,2} = 0.01, t_{max} = D+1, I_{max,2} = 1e-6$)

[b] RBFI–ESS–SANS–BTRS–GIS–ETS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5, I_{max} = 1e-8, \Delta_{min,2} = 0.01$)

6.6.3 Comparisons between RESBGEFS and various metaheuristic algorithms on 24 benchmark problems

In Section 3.6.5, IATRO – the first proposed algorithm in this thesis is compared with various metaheuristic algorithms which conduct the optimization process by computing values of the functions directly. As RESBGEFS has experienced lots of significant developments as compared to the IATRO and is the final proposed algorithm in this thesis, it is necessary to compare RESBGEFS with these direct methods again, as well as IATRO to see how much difference has been made.

As shown in Table 6.9, the ENFEs values of RESBGEFS on 24 benchmark problems are compared with the results obtained by others. Since the smallest ENFEs value among all the results is marked in bold face, it is quite clear that RESBGEFS outperforms other algorithms a lot on 23 out of the 24 problems except on G12. But as described in Section 3.6.5, the results of NSES [87] in Table 6.9 are the number of function evaluations when the optimization process finds a feasible solution of which the objective function value is within $1e-4$ from the best-known optimum. They are not the true ENFEs values when the optimization process actually converges without any prior knowledge of the known optimum like what RESBGEFS does. Thus, the fact that the 'ENFEs' of NSES on G12 is smaller than the ENFEs value of RESBGEFS can just indicate that NSES is competitive in solving G12 but does not mean that NSES is superior to RESBGEFS in tackling this problem.

Nevertheless, from the average ENFEs value (AENFEs), the advantages of RESBGEFS in solving these 24 benchmark problems are more distinct. RESBGEFS, rank-iMDDE and LCA are three algorithms that are able to solve all these problems but the AENFEs of RESBGEFS is only 324, about 1/162 and 1/933 of the value obtained by rank-iMDDE and LCA respectively. COPSO can not solve G21 and G23 and the AENFEs value of the 21 problems is 2260 times larger than the result of RESBGEFS. The optimization performance of NSES is close to Q-COM and they both require over 40 times the number of function evaluations of RESBGEFS to solve the 21 problems. For IATRO, which is the first presented algorithm in the thesis, fails to solve 6 out of 24 problems and the AENFEs value of the rest 18 problems is about 8 times as much as the value of RESBGEFS.

In conclusion, RESBGEFS only requires several tenths to several thousandths of the function evaluations used by the direct metaheuristic algorithms to accomplish global optimization on low-dimensional CBO problems. For problems that each function evaluation is time-consuming or

Table 6.9: Comparisons among RESBGEFS, IATRO and various direct algorithms on 24 benchmark problems in terms of the efficient number of function evaluations (ENFEs)

Prob.	RESBGEFS	IATRO	rank-iMDDE [152]	LCA [113]	COPSO [106]	NSES [87]	Q-COM [151]
G01	**313**	1802	80483	225000	95397	31710	15520
G03	**470**	918	49572	225000	315123	19534	18618
G04	**93**	124	31649	225000	65087	5357	15066
G05	**74**	4549	33615	225000	315257	1558	15117
G06	**44**	N.S.[a]	12942	225000	53410	732	15039
G07	**334**	698	62276	225000	233400	171990	15274
G08	**350**	973	2961	225000	6470	541	12762
G09	**332**	1943	24849	225000	79570	9357	15407
G10	**232**	N.S.[a]	92718	225000	224740	85623	15782
G11	**105**	657	7340	225000	315000	135	15015
G12	357	6241	3101	225000	6647	**212**	469
G13	**475**	N.S.[a]	38988	225000	315547	3103	16137
G14	**841**	1886	127553	500000	9807000	6093	16061
G15	**120**	6181	19067	500000	315100	757	15128
G16	**91**	229	18527	500000	40960	8982	15364
G17	**656**	N.S.[a]	64539	500000	412968	3203	50556
G18	**361**	1687	60084	500000	185654	3353	15393
G19	**1139**	N.S.[a]	181296	500000	566601	56681	15241
G21	**664**	N.S.[a]	89617	500000	N.S.[a]	46722	16167
G23	**98**	15188	205337	500000	N.S.[a]	9757	15458
G24	**100**	237	5490	500000	19157	161	15028
WBD	**120**	412	15000	15000	30000	N.A.[b]	N.A.[b]
SPD	**291**	4005	10000	15000	30000	N.A.[b]	N.A.[b]
SRD	**121**	707	19920	24000	N.A.[b]	N.A.[b]	N.A.[b]
$\frac{\text{AENFEs}}{\text{AENFEs of RESBGEFS}}$	$\frac{324}{324} = 1$	$\frac{2691}{324} \approx 8$	$\frac{52372}{324} \approx 162$	$\frac{302250}{324} \approx 933$	$\frac{741239}{328} \approx 2260$	$\frac{17966}{345} \approx 52$	$\frac{14378}{345} \approx 42$

[a] The result is not available because the algorithm fails to solve this problem.

[b] The result is not available in the corresponding paper.

[*] The result in bold font means that it is the smallest ENFEs value among all results.

96

even money-consuming, RESBGEFS is one of the best practical optimization method.

6.6.4 Comparisons among RESBGEFS, IATRO and various metamodel-based algorithms on 26 benchmark problems

Similar to comparing IATRO with state-of-the-art metamodel-based algorithms in Section 3.6.6, RESBGEFS is compared with these metamodel-based algorithms again as well as IATRO, as shown in Table 6.10 to illustrate the advantages of RESBGEFS. As pointed out in Section 3.6.6, the results of other algorithms (COBRA, eDIRECT-C, SADE-kNN, SACOBRA, SCGOSR) can not reflect the true performance in optimizing CBO problems for two main reasons. First, the majority of metamodel-based algorithms aim to obtaining an improved feasible solution rather than finding the global optimum and the optimization statistics such as the SR values and FR values were not reported in details in their papers. So it is difficult to compare RESBGEFS with these algorithms in terms of ENFEs values. Instead, the EAREs values of RESBGEFS which describes how many function evaluations have been used to find the best solution in the optimization process would be more appropriate for making comparisons between RESBGEFS and other metamodel-based algorithms. Second, the termination criterion used by other metamodel-based algorithms are related to the known global optimum of the test problem, which is usually unavailable in solving real-world optimization problems. Thus, only the results of RESBGEFS as well as IATRO can be considered to demonstrate the true performance in solving real-world CBO problems.

From Table 6.10, although RESBGEFS runs in more strict conditions, the superiorities of RESBGEFS are distinct in all aspects. For all the test problems, the objective function values obtained by RESBGEFS are the best or equally best among all the results. Moreover, among the algorithms which can obtain the best solution of a problem, RESBGEFS only requires a smaller number of function evaluations as compared to other algorithms. There are only five problems that RESBGEFS might not be the best algorithm in solving them. For G01 and G14, although the EAREs values of IATRO are the smallest, it does not mean that IATRO is better than RESBGEFS in solving these two problems. Actually, this can just indicate that IATRO converges faster than RESBGEFS. Considering the ENFEs values as shown in Table 6.9, RESBGEFS is far superior to IATRO in solving these two problems in terms of efficiency and stability. For G07, G11

Table 6.10: Comparisons among RESBGEFS, IATRO and metamodel-based algorithms on 26 benchmark problems

Prob.	Criteria	RESBGEFS	IATRO	COBRA [115]	eDIRECT-C [118]	SADE-kNN [153]	SACOBRA[a][116]	SCGOSR [74]
G01	Best	**-15.0000**	**-15.0000**	≤ -14.85	-14.9998	**-15.0000**	-15.0	N.A.
	EAREs	209	**138**	> 387	147	> 3722	100	N.A.
G02	Best	**-0.7931**	-0.3812	N.A.	-0.2480	-0.7429	-0.3466	N.A.
	EAREs	**1232**	1237	N.A.	> 1000	N.A.	400	N.A.
G03	Best	**-1.0005**	**-1.0005**	≤ -0.33	-0.9989	-0.4515	-1.0	N.A.
	EAREs	**413**	705	> 451	145	N.A.	300	N.A.
G04	Best	**-30665.5394**	**-30665.5396**	N.A.	**-30665.5385**	**-30665.5386**	-30665.539	-31026
	EAREs	**28**	78	N.A.	65	2598	200	54
G05	Best	**5126.4981**	**5126.4981**	≤ 5150	5145.8149	5126.49	5126.498	N.A.
	EAREs	**43**	351	13	413	> 17810	200	N.A.
G06	Best	**-6961.8139**	-6961.8128	≤ -6800	**-6961.8137**	**-6961.8138**	-6961.81	-6961.8
	EAREs	**24**	294	53	35	1235	100	79
G07	Best	**24.3062**	**24.3062**	≤ 25	**24.3062**	24.3073	24.306	24.3149
	EAREs	330	653	199	**152**	N.A.	200	178
G08	Best	**-0.0958**	**-0.0958**	≤ -0.09	**-0.095822**	**-0.09582**	**-0.0958**	**-0.0958**
	EAREs	**60**	148	30	154	292	200	52
G09	Best	**680.6301**	**680.6301**	≤ 1000	785.6795	680.638	680.761	826.30
	EAREs	**291**	607	> 275	> 1000	N.A.	300	116
G10	Best	**7049.2480**	7049.2515	≤ 8000	7049.2484	7049.249	7049.253	N.A.
	EAREs	**194**	902	276	105	N.A.	300	N.A.
G11	Best	**0.7500**	**0.7500**	N.A.	**0.7499**	**0.7499**	0.75	N.A.
	EAREs	59	114	N.A.	**33**	> 2995	100	N.A.
G12	Best	**-1.0000**	**-1.0000**	N.A.	**-1.0000**	**-1.0000**	N.A.	N.A.
	EAREs	58	154	N.A.	**52**	> 386	N.A.	N.A.
G13	Best	**0.0539**	0.0541	N.A.	0.6472	**0.05394**	N.A.	N.A.
	EAREs	**166**	245	N.A.	> 1000	> 43907	N.A.	N.A.
G14	Best	**-47.7611**	**-47.7611**	N.A.	N.A.	-47.764	N.A.	N.A.
	EAREs	711	**601**	N.A.	N.A.	> 55179	N.A.	N.A.
G15	Best	**961.7152**	**961.7152**	N.A.	N.A.	**961.7150**	N.A.	N.A.
	EAREs	**68**	228	N.A.	N.A.	> 11431	N.A.	N.A.
G16	Best	**-1.9052**	**-1.9052**	≤ -1.8	N.A.	-1.9051	N.A.	N.A.
	EAREs	**41**	202	38	N.A.	4633	N.A.	N.A.
G17	Best	**8853.5401**	-	N.A.	N.A.	8853.53	N.A.	N.A.
	EAREs	**333**	-	N.A.	N.A.	> 69887	N.A.	N.A.
G18	Best	**-0.8660**	**-0.8660**	≤ -0.8	N.A.	-0.8654	N.A.	N.A.
	EAREs	**156**	750	> 196	N.A.	> 253743	N.A.	N.A.
G19	Best	**32.6556**	36.2503	≤ 40	N.A.	32.6632	N.A.	N.A.

continued

Table 6.10 – *continued*

Prob.	Criteria	RESBGEFS	IATRO	COBRA [115]	eDIRECT-C [118]	SADE-kNN [153]	SACOBRA[a][116]	SCGOSR [74]
	EAREs	**1062**	1910	698	N.A.	N.A.	N.A.	N.A.
G21	Best	**193.7869**	258.6453	N.A.	N.A.	193.7546	N.A.	N.A.
	EAREs	**250**	514	N.A.	N.A.	N.A.	N.A.	N.A.
G23	Best	**-400.0000**	**-400.0000**	N.A.	N.A.	-400.055	N.A.	N.A.
	EAREs	**53**	1186	N.A.	N.A.	$>$ 68852	N.A.	N.A.
G24	Best	**-5.5080**	**-5.5080**	≤ -5	N.A.	**-5.5080**	N.A.	N.A.
	EAREs	**23**	76	9	N.A.	765	N.A.	N.A.
WBD	Best	**1.7249**	**1.7249**	≤ 2.5	N.A.	N.A.	N.A.	**1.7249**
	EAREs	**99**	408	165	N.A.	N.A.	N.A.	102
SPD	Best	**0.012666**	0.0126720	N.A.	**0.012666**	N.A.	N.A.	0.01267
	EAREs	**223**	768	N.A.	292	N.A.	N.A.	76
PVD	Best	**6059.7143**	7367.6123	N.A.	N.A.	N.A.	N.A.	N.A.
	EAREs	**78**	243	N.A.	N.A.	N.A.	N.A.	N.A.
SRD	Best	**2994.4702**	**2994.4711**	N.A.	N.A.	N.A.	N.A.	N.A.
	EAREs	**61**	556	N.A.	N.A.	N.A.	N.A.	N.A.

[a] In the paper of SACOBRA, only median statistics are provided.

[*] The objective function value in bold face means that it is the best or equally best value among all the results obtained by these algorithms. And the best EARs of the problem obtained by an algorithm will be marked in bold face only if this algorithm can obtain the best solution.

and G12, eDIRECT-C is slight better than RESBGEFS but it is far inferior to RESBGEFS in solving the another 11 out of 14 test problems since it can not obtain the corresponding global optima.

In conclusion, RESBGEFS is much better than other metamodel-based algorithms in terms of efficiency, robustness and accuracy. Moreover, it is currently one of the few metamodel algorithms that can efficiently converge to a feasible solution of the CBO problem without the prior knowledge of the global optimum.

6.7 Summary

In this chapter, the trust region based and surrogate-assisted optimization framework (RATRO) is enhanced with the fast computation strategy (FCS) and the successive refinement strategy (SRS) and the new optimization framework is called RATRLO (RBF-assisted and trust region based large-scale optimization framework). More specifically, FCS is an implementation of the advanced fast-computation modules in the Python environment and SRS is the novel strategy and is the key to the accuracy of the optimal solutions. As shown in Algorithm 9, it aims at tackling large-scale CBO problems, for example the MOPTA08 problem which has 124 design variables and 68 inequality constraints. Usually, one iteration of RESBGE involves one DOE process, one metamodel building process by RBF interpolation and one run of the SQP process to solve the approximate subproblem. This can take a number of minutes (up to 30 minutes in worst cases) but with the Numba support and the parallel computation technique, the computational time can decrease to about 10 seconds, which enables the development of new strategies for solving large-scale optimization problems. And the successive refinement strategy (SRS) is the key to obtain an improved feasible solution of the MOPTA08 problem is. As shown in Algorithm 8, by iteratively updating the metamodels and refining the solution of the subproblem, a good sub-optimal solution can be generated. The experimental results of RESBGEFS on the MOPTA08 problem prove that RESBGEFS is among the best metamodel-based algorithms in solving this problem. And due to the remarkable less computational time of RESBGEFS (about 30 minutes for solving the MOPTA08 after 1000 function evaluations) as compared to other algorithms (for example, ConstrLMSRBF takes about 9 hours to solving the MOPTA08 problem after 2000 function evaluations), RESBGEFS has more potentials in further development for solving large-scale, severely-constrained black-box

Algorithm 9: Framework of RATRLO

Input:

See Table 5.2 and Table 6.3.

Output:

$\boldsymbol{x}_{opt}$: The best point encountered through the optimization process.

for k **in** $[0, K_{max}]$ **do**

Step 1. Design of Experiments (DOE) ▷ See Algorithm 6

$n^k, X^k = \texttt{DOE}(\boldsymbol{x}^k, k, K_{max}, \boldsymbol{A}^k, \boldsymbol{B}^k, \Delta_{ext}, N_{plan}, X_{all})$

Step 2. Black-box Evaluations ▷ Parallel computation

Objective function evaluations: $f(X^k) \subseteq \mathbb{R}^{n^k}$.

Constraint function evaluations: $g_j(X^k)\,(j = 1, \ldots, m) \subseteq \mathbb{R}^{n^k}$

Step 3. Self-adaptive Normalization ▷ See Algorithm 4

$f(\boldsymbol{x}), g_j(\boldsymbol{x})\,(j = 1, \ldots, m) =$
$\texttt{SANS}(\delta_f, \delta_g, f(X^k), g_j(X^k)\,(j = 1, \ldots, m))$

Step 4. Successive Refinement Strategy ▷ See Algorithm 8

$\boldsymbol{x}^{k+1} = \texttt{SRS}(\boldsymbol{x}^k, k, X^k, \boldsymbol{f}(X^k), \boldsymbol{g}(X^k), \boldsymbol{A}^k, \boldsymbol{B}^k, t_{max})$

Step 5. Moving Trust Region Strategy

if *Termination not satisfied* **then**

▷ Update move limits by any trust region strategy, for example BTRS (Section 5.2)

$Q^{k+1} : [\boldsymbol{A}^{k+1}, \boldsymbol{B}^{k+1}]$

$k = k + 1$ ▷ Increment the iteration number

else

Traverse all the points.

Return $\boldsymbol{x}_{opt}$.

optimization problems within limited computational budget.

Besides, the results of RESBGEFS on 24 low-dimensional benchmark problems (except G02 and PVD) show that about 1/3 of the function evaluations can be reduced as compared to the results of RESBGE due to the high cost-effectiveness of SRS. On average, RESBGEFS only requires 324 function evaluations to absolutely obtain the global optima of these 24 problems. Compared with state-of-the-art direct algorithms and metamodel-based algorithms, RESBGEFS is among the best if not the best algorithm to solve these problems with high efficiency, accuracy and robustness. Meanwhile, RESBGEFS is also one of the few metamodel-based algorithms that possesses good global convergency, which is a valuable contribution to the area of metamodel-based optimization.

7

Concluding remarks and recommendations

7.1 Summary of findings, discussions and conclusions

Multidisciplinary design optimization based on computation-intensive simulations of complex systems is becoming increasingly popular in various engineering subjects. As each simulation could be expensive in terms of elapsed execution time and/or computational cost, only a limited number of simulations are allowed. This issue is particularly acute when design optimization is considered because all optimization methods, either nature-inspired algorithms or deterministic optimization algorithms require a great deal of function evaluations to find an optimal design. One popular and practical way to address this issue is to apply metamodels which approximate the time-consuming simulations with simpler analytical models. Then, various optimization methods can be performed on those models to search for the optimum, which is referred as metamodel-based design optimization (MBDO). Although MBDO has attracted widespread attention and has been under continuous development, the state of the art about computationally expensive constrained optimization is less advanced [61, 81]. To tackle this challenge in solving expensive constrained black-box optimization problems, a RBF-assisted and trust region based large-scale optimization (RATRLO) framework has been developed in this thesis.

In the first place, an intrinsically linear function assisted and trust region based optimization framework (IATRO) has been established for solving low-dimensional CBO (constrained black-box optimization) problems. Generally, IATRO adopts strong points of the multipoint approximation method (MAM) [77, 135, 167] which replaces the original complex optimization problem by a succession of simpler mathematical sub-problems

in a series of trust regions where each objective and constraint functions are approximated by ILF (intrinsically linear function). The solution of an individual sub-problem becomes the starting point in the next iteration and then, the move limits are updated and the optimization loop is repeated until the optimum is reached. The main contributions of IATRO developed in Python environment include the modified weighted least squares regression subroutine and SQP solver, the progress curve and debug report, the improvement of the output solution, the random starting point generator, the benchmark functionality and a collection of benchmark CBO problems. Although the original MAM can hardly solve a complex CBO problem, IATRO proves to be a competitive optimization method as compared to several state-of-the-art direct metaheuristic algorithms and metamodel-based optimization algorithms. Through numerical experiments of 26 benchmark problems (22 G-problems, 2 continuous engineering design problems and 2 mix-variables engineering design problems), results show that 18 problems can be solved by IATRO within an acceptable number of function evaluations but the another eight problems (G02, G06, G10, G13, G17, G19, G21 and PVD) are difficult to address.

In order to enhance the optimization performance of IATRO, several strategies have been further developed including the economical sampling strategy (ESS), the self-adaptive normalization strategy (SANS) and the modified trust region strategy (MTRS). More specifically, ESS enables IATRO to reduce about 20% of the required number of function evaluations while keeping the global searching ability; SANS brings remarkable advantages in solving CBO problems (G01, G06, G10 and G17), in which the values of constraint functions range across different orders of magnitude; and MTRS leads to the optimization process more likely moving to the right direction as the average success rate of IEM (IATRO–ESS–MTRS) in the 26 benchmark problems is 48%, while the average success rate of 36% of IE (IATRO–ESS) can be observed. Although IEMS (IATRO–ESS–MTRS–SANS) is capable of robustly obtaining feasible, global solutions to most benchmark problems, there are ten problems (G01, G02, G05, G10, G12, G13, G15, G17, G21 and PVD) that can not be solved for the converged solutions. This requires a more powerful algorithm to be developed.

Through in-depth research on the statistical results of the above then test problems using IEMS, it is found that one reason for the instability of obtaining the true optima is the inadequate approximations built by the intrinsically linear functions (ILF). Therefore, the radial basis function

assisted optimization framework (RATRO) has been further developed in this research. Generally, the RBF interpolation is far superior to the intrinsically linear approximation in terms of accuracy so that the average success rate of RESM (RBFI–ESS–SANS–MTRS) in the 26 problems reaches 69.38%, which is 17 percentage points more than the value of IEMS (52.15%). Six out of the ten problems which are extremely difficult to IEMS can be solved by RESM with a success rate of over 24% and the remaining four unsolved problems are G02, G10, G21 and PVD. In order to tackling these problems, a number of novel strategies have been proposed. First, a balanced moving trust region strategy (BTRS) has been developed to achieve a good tradeoff between exploration and exploitation of the search space for the optimal solution. With the trust region strategy (BTRS) in RESM, the enhanced algorithm has been called RESB (RBFI–ESS–SANS–BTRS). Results show that RESB has the capability to solve the G21 problem with a high success rate over 48%. It is also noted that BTRS remarkably alleviates the difficulty in optimizing the problem of G01 as the success rate nearly quadruples from 24% to 88%. Moreover, the global intelligence selection (GIS) has been proposed, which adds additional sampling points that are closest to the starting point and outside the current trust region into the fitting set. This can not only refine the approximations in the current search subregion but also help the optimization process escape from the local optima. Therefore, RESBG (RBFI–ESS–SANS–BTRS–GIS) is able to solve the 24 benchmark problems (except G02 and PVD) with a success rate of 81.87% on average. Last but not the least, with the early termination strategy (ETS) which enables the optimization process to abort appropriately when there is little room improving the solution, RESBGE (RBFI–ESS–SANS–BTRS–GIS–ETS) can reduce the number of function evaluations by 14.5% to obtain the global optimum with a slight decrease in success rate, as compared to RESBG. G02 and PVD are two remaining unsolved problems but both RESBG and RESBGE can definitely find near-optimal solutions. Since G02 is a twenty-dimensional optimization problem with a periodic and multimodal objective function, currently there is no metamodel-based algorithm that could solve it with high accuracy and efficiency. As PVD belongs to a mixed continuous/discrete variable problem, how to improve the performance of the algorithm to solve this problem is considered future work.

Finally, the optimization framework of RATRO (radial basis function assisted and trust region based optimization) are expanded to solve large-scale optimization problems with the development of the fast computation strategy (the Numba compiler and parallel computation

technique) and the successive refinement strategy (SRS). The MOPTA08 problem [119] which has 124 design variables and 68 inequality constraints is used as a benchmark to test whether the RATRO has the capabilities to solve this problem with severely limited computational budget. Most of the surrogate-based algorithms suffer from the time-consuming optimization process when a large number of design variables are involved. Usually one basic optimization process of RESBGE in solving the MOPTA08 problem including one DOE process, one metamodel building process by RBF interpolation and one run of the SQP process to solve the approximate subproblem, takes 5-30 minutes. With advances in the Numba compiler which forces the Python functions to run at native machine code speed, and the parallel computation technique to execute multiple function evaluations simultaneously, the average computational time of this basic process can be dropped to about 10 seconds. Besides, the SRS procedure aiming at obtaining a sufficiently optimal solution in each iteration by successively refining the metamodels proves to be an efficient and practical approach in solving high-dimensional CBO problems. For the MOPTA08 problem, RESBGEFS can achieve about 90% of the potential reduction on the objective function value within only 1000 function evaluations, which can be considered a state of the art. Furthermore, the SRS procedure also works remarkably in solving low-dimensional CBO problems. Averagely, RESBGEFS just uses 324 function evaluations that is just two thirds of the total number required by RESBGE to obtain the global optimum of one benchmark problem (except G02 and PVD). Moreover, RESBGEFS is far superior to RESBGE on optimizing G02 as an optimal solution with the objective function value -0.7931 can be found, which is very close to the global optimum (-0.8036).

In conclusions, the main contributions include the reconstitution of multipoint approximation method (MAM) in Python environment, the implementation of the radial basis function interpolation (RBFI), the utilization of the fast-computation modules (FCS), the development of the economical sampling strategy (ESS), the self-adaptive normalization strategy (SANS), the global intelligence selection strategy (GIS), the balanced trust region strategy (BTRS), the early termination strategy (ETS) and the successive refinement strategy (SRS). And the final developed optimization framework – RATRLO (RBF-assisted and trust region based large-scale optimization) belongs to the best metamodel-based optimization schemes for solving complex constrained black-box problems. Regardless of whether the optimization problem belongs to low-dimensional or high-dimensional, linear or nonlinear, continuous or mixed and equality constrained or inequality

constrained, RATRLO is able to find the global optimum or a satisfying feasible solution within limited computational budget. Furthermore, RATRLO exhibits the global convergence for the searching process of optimal solutions and provides a useful insight into the development of advanced optimization methods, e.g., metaheuristic technique, artificial neural network, for solving complex constrained black-box optimization problems with high levels of robustness, efficiency and accuracy.

7.2 Future work

Although RATRLO has gained huge success in solving constrained black-box optimization problems, the potentials of this trust-region based searching scheme have not yet been fully realized.

Based on the numerical results presented earlier, G02 and PVD are the two most challenging problems. How to stably solve these kinds of problems is part of the future work. G02 is a twenty-dimensional optimization problem with a periodic and multimodal objective function. The key to solving this type of problem is to force the optimization process to escape from the local optima. As RATRLO shows advantages in converging to a local optimum and nature-inspired algorithms are good at finding the global optimum, a hybrid algorithm can be developed, this is to say, metaheuristic algorithms should be investigated in the main optimization framework where RATRLO is applied to accelerate the convergence of local search. This metaheuristic algorithm might work well in solving problems with numerous local optima for the global solution.

PVD and SRD are two mixed continuous/discrete variable problems. Although RATRLO works well in solving SRD, the performance is not sufficiently good in optimizing PVD. Results show that RESBGEFS usually obtains a sub-optimum close to the global optimum. This is because RESBGEFS does not deal with mixed variables well. Actually, in the DOE process all the variables have been considered continuous values and only the evaluation process pays attention to whether the variable is continuous or discrete. This results in the inconsistency between the design vector (continuous property) and the function values (evaluated by mixed variables). Therefore, specific approaches have to be developed for properly handling mixed variables. For example, the variables have to be clearly classified by specific algorithms as continuous and discrete groups so that the correct design vectors can be generated in the sampling process. The moving

trust region strategy should be modified accordingly. It is believed that mixed-variable problems can be accurately solved once the strategy for searching subregions of discrete variables can be developed.

Moreover, as state-of-the-art metamodels and optimization solvers can be easily implemented in RATRLO, it needs further research to evaluate the impact of these advances on the performance of RATRLO. In current implementation, the cubic RBF with a polynomial tail is good for fitting the metamodel. Other forms of RBF and metamodels (For example, Kriging) are worthwhile to be tested and compared. Meanwhile, as a solver can be effective in solving a group of problems but not appropriate on many others, another part of the future work will aim at looking for an intelligent way to adaptively select the internal optimizer.

Last but not the least, more high-dimensional and real-world benchmarks should be tested by the current optimization framework so that it can provide a better insight into the existing weakness in global searching and the development of more robust and accurate optimization framework in the future of work. It will provide a better insight into the existing weaknesses in global searching and support further improvements.

Bibliography

[1] David H. Wolpert and William G. Marcready. "No-Free-Lunch Theorem". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82.

[2] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. New York: Springer, 2006.

[3] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2009.

[4] John H. Holland. *Adaptation in Natural and Artificial Systems: An introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1975, p. 183.

[5] J Kennedy and R Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. IEEE, 1995, pp. 1942–1948.

[6] Marco Dorigo and Gianni Di Caro. "Ant Colony Optimization: A New Meta-Heuristic". In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 2. Washington, DC, 1999, p. 1477.

[7] Patrick N. Koch et al. "Statistical approximations for multidisciplinary design optimization: The problem of size". In: *Journal of Aircraft* 36.1 (1999), pp. 275–286.

[8] David E Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Vol. Addison-We. 1989, p. 432.

[9] TImothy W. Simpson et al. "Metamodels for Computer-Based Engineering Design: Survey and Recommendations". In: *Engineering with Computers* 17.2 (2001), pp. 129–150.

[10] Nestor V. Queipo et al. "Surrogate-based analysis and optimization". In: *Progress in Aerospace Sciences* 41.1 (2005), pp. 1–28.

[11] J. -F. M. Barthelemy and R. T. Haftka. "Approximation concepts for optimum structural design — a review". In: *Structural optimization* 5.3 (1993), pp. 129–144.

[12] Raphael T. Haftka, Elaine P. Scott, and Juan R. Cruz. "Optimization and Experiments: A Survey". In: *Applied Mechanics Reviews* 51.7 (1998), pp. 435–448.

[13] G. Gary Wang and S. Shan. "Review of Metamodeling Techniques in Support of Engineering Design Optimization". In: *Journal of Mechanical Design* 129.4 (2007), p. 370.

[14] Raymond H. Myers, Douglas C. Montgomery, and Christine M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. 4th. 2016.

[15] R. Jin, W. Chen, and T. W. Simpson. "Comparative studies of metamodelling techniques under multiple modelling criteria". In: *Structural and Multidisciplinary Optimization* 23.1 (2001), pp. 1–13.

[16] Ann-britt Ryberg, Rebecka Domeij Bäckryd, and Nilsson Larsgunnar. *Metamodel-Based Multidisciplinary Design Optimization for Automotive Applications*. September. 2012.

[17] M. D. McKay, R. J. Beckman, and W. J. Conover. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". In: *Technometrics* 21.2 (1979), pp. 239–245.

[18] M. E. Johnson, L. M. Moore, and D. Ylvisaker. "Minimax and maximin distance designs". In: *Journal of Statistical Planning and Inference* 26.2 (1990), pp. 131–148.

[19] J. M. Hammersley. "MONTE CARLO METHODS FOR SOLVING MULTIVARIABLE PROBLEMS". In: *Annals of the New York Academy of Sciences* 86.3 (1960), pp. 844–874.

[20] Kai Tai Fang et al. "Uniform design: Theory and application". In: *Technometrics* 42.3 (2000), pp. 237–248.

[21] A. B. Owen. "Orthogonal Arrays for Computer Experiments, Integration and Visualization". In: *Statistica Sinica* 2 (1992), pp. 439–452.

[22] J. R. Koehler and A. B. Owen. "Design and Analysis of Experiments". In: *Handbook of Statistic*. Ed. by S. Ghosh and C. Rao. Vol. 13. North-Holland, 1996, pp. 261–308.

[23] Timothy W. Simpson, Dennis K. J. Lin, and Wei Chen. "Sampling Strategies for Computer Experiments: Design and Analysis". In: *International Journal of Reliability and Applications* 2.3 (2001), pp. 209–240.

[24] R H Myers et al. "Response Surface Methodology: A Retrospective and Literature Survey". In: *Journal of Quality Technology* 36.1 (2004), pp. 53–78.

[25] Nira Dyn, David Levin, and Samuel Rippa. "Numerical procedures for surface fitting of scattered data by radial functions". In: *SIAM Journal on Scientific and Statistical Computing* 7.2 (1986), pp. 639–659.

[26] Jerome Sacks et al. "Design and Analysis of Computer Experiments". In: *Statistical Science* 4.4 (Nov. 1989), pp. 409–423.

[27] Jerome H. Friedman. "Multivariate Adaptive Regression Splines". In: *The Annals of Statistics* 19.1 (1991), pp. 1–67.

[28] David E. Rumelhart, Bernard Widrow, and Michael A. Lehr. "The Basic Ideas in Neural Networks". In: *Communications of the ACM* 37.3 (1994), pp. 87–92.

[29] Stella M. Clarke, Jan H. Griebsch, and Timothy W. Simpson. "Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses". In: *Transactions of ASME, Journal of Mechanical Design* 127.6 (2005), pp. 1077–1087.

[30] Byeong Soo Kim, Yong Bin Lee, and Dong Hoon Choi. "Comparison study on the accuracy of metamodeling technique for non-convex functions". In: *Journal of Mechanical Science and Technology* 23.4 (2009), pp. 1175–1181.

[31] Michael T.M. Emmerich, Kyriakos C. Giannakoglou, and Boris Naujoks. "Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels". In: *IEEE Transactions on Evolutionary Computation* 10.4 (2006), pp. 421–439.

[32] Jian Wen et al. "Energy and cost optimization of shell and tube heat exchanger with helical baffles using Kriging metamodel based on MOGA". In: *International Journal of Heat and Mass Transfer* 98 (2016), pp. 29–39.

[33] Zongzhao Zhou et al. "Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization". In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 37.1 (Jan. 2007), pp. 66–76.

[34] Bo Liu, Qingfu Zhang, and Georges G. E. Gielen. "A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems". In: *IEEE Transactions on Evolutionary Computation* 18.2 (Apr. 2014), pp. 180–192.

[35] Dan Guo et al. "Heterogeneous Ensemble-Based Infill Criterion for Evolutionary Multiobjective Optimization of Expensive Problems". In: *IEEE Transactions on Cybernetics* 49.3 (2019), pp. 1012–1025.

[36] C. Praveen and R. Duvigneau. "Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design". In: *Computer Methods in Applied Mechanics and Engineering* 198.9-12 (2009), pp. 1087–1096.

[37] Rommel G. Regis. "Particle swarm with radial basis function surrogates for expensive black-box optimization". In: *Journal of Computational Science* 5.1 (Jan. 2014), pp. 12–23.

[38] Chaoli Sun et al. "Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems". In: *IEEE Transactions on Evolutionary Computation* 21.4 (2017), pp. 644–660.

[39] Haibo Yu et al. "A generation-based optimal restart strategy for surrogate-assisted social learning particle swarm optimization". In: *Knowledge-Based Systems* 163 (2019), pp. 14–25.

[40] Leonardo G. Fonseca, Afonso C.C. Lemonge, and Helio J.C. Barbosa. "A study on fitness inheritance for enhanced efficiency in real-coded genetic algorithms". In: *2012 IEEE Congress on Evolutionary Computation, CEC 2012* June (2012).

[41] Xinjing Wang et al. "A Novel Evolutionary Sampling Assisted Optimization Method for High-Dimensional Expensive Problems". In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 815–827.

[42] Xiwen Cai et al. "Surrogate-guided differential evolution algorithm for high dimensional expensive problems". In: *Swarm and Evolutionary Computation* 48.May 2018 (2019), pp. 288–311.

[43] M. D. Parno, T. Hemker, and K. R. Fowler. "Applicability of surrogates to improve efficiency of particle swarm optimization for simulation-based problems". In: *Engineering Optimization* 44.5 (May 2012), pp. 521–535.

[44] Yuanfu Tang, Jianqiao Chen, and Junhong Wei. "A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions". In: *Engineering Optimization* 45.5 (May 2013), pp. 557–576.

[45] Kambiz Haji Hajikolaei et al. "Surrogate-assisted Self-accelerated Particle Swarm Optimization". In: *10th AIAA Multidisciplinary Design Optimization Conference*. January. Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2014.

[46] ALAIN RATLE. "Kriging as a surrogate fitness landscape in evolutionary optimization". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 15.1 (Jan. 2001), pp. 37–49.

[47] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. "A framework for evolutionary optimization with approximate fitness functions". In: *IEEE Transactions on Evolutionary Computation* 6.5 (2002), pp. 481–494.

[48] Holger Ulmer, Felix Streichert, and Andreas Zell. "Evolution strategies assisted by Gaussian processes with improved preselection criterion". In: *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.* Vol. 1. July 2015. IEEE, 2003, pp. 692–699.

[49] D. Buche, N.N. Schraudolph, and Petros Koumoutsakos. "Accelerating Evolutionary Algorithms With Gaussian Process Fitness Function Models". In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 35.2 (May 2005), pp. 183–194.

[50] Ava Shahrokhi and Alireza Jahangirian. "A surrogate assisted evolutionary optimization method with application to the transonic airfoil design". In: *Engineering Optimization* 42.6 (2010), pp. 497–515.

[51] Aalae Benki, Abderrahmane Habbal, and Gael Mathis. "A metamodel-based multicriteria shape optimization process for an aerosol can". In: *Alexandria Engineering Journal* 57.3 (2018), pp. 1905–1915.

[52] Haibo Yu et al. "Surrogate-assisted hierarchical particle swarm optimization". In: *Information Sciences* 454-455 (2018), pp. 59–72.

[53] Xiwen Cai et al. "An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems". In: *Knowledge-Based Systems* 184 (2019), p. 104901.

[54] X. Cai, L. Gao, and Fan Li. "Sequential approximation optimization assisted particle swarm optimization for expensive problems". In: *Applied Soft Computing Journal* 83 (2019), p. 105659.

[55] Yew S Ong, Prasanth B Nair, and Andrew J Keane. "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling". In: *AIAA JOURNAL* 41.4 (2003).

[56] Dudy Lim et al. "Generalizing surrogate-assisted evolutionary computation". In: *IEEE Transactions on Evolutionary Computation* 14.3 (2010), pp. 329–355.

[57] Xiaoyan Sun, Dunwei Gong, and Wei Zhang. "Interactive genetic algorithms with large population and semi-supervised learning". In: *Applied Soft Computing Journal* 12.9 (2012), pp. 3004–3013.

[58] Minh Nghia Le et al. "Evolution by adapting surrogates". In: *Evolutionary Computation* 21.2 (2013), pp. 313–340.

[59] Zan Yang et al. "A surrogate-assisted particle swarm optimization algorithm based on efficient global optimization for expensive black-box problems". In: *Engineering Optimization* 51.4 (Apr. 2019), pp. 549–566.

[60] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13 (1998), pp. 455–492.

[61] Raphael T. Haftka, Diane Villanueva, and Anirban Chaudhuri. "Parallel surrogate-assisted global optimization with expensive functions – a survey". In: *Structural and Multidisciplinary Optimization* 54.1 (2016), pp. 3–13.

[62] Mohamed Amine Bouhlel et al. "Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method". In: *Engineering Optimization* (2018), pp. 1029–0273.

[63] Andreas Krause and Cheng Soon Ong. "Contextual Gaussian process bandit optimization". In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011* May (2011).

[64] Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. "An informational approach to the global optimization of expensive-to-evaluate functions". In: *Journal of Global Optimization* 44.4 (2009), pp. 509–534.

[65] Philipp Hennig and Christian J. Schuler. "Entropy search for information-efficient global optimization". In: *Journal of Machine Learning Research* 13 (2012), pp. 1809–1837.

[66] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. "Predictive entropy search for efficient global optimization of black-box functions". In: *Advances in Neural Information Processing Systems* 1.January (2014), pp. 918–926.

[67] Rommel G. Regis and Christine A. Shoemaker. "Constrained global optimization of expensive black box functions using radial basis functions". In: *Journal of Global Optimization* 31.1 (2005), pp. 153–171.

[68] Liqun Wang, Songqing Shan, and G. Gary Wang. "Mode-pursuing sampling method for global optimization on expensive black-box functions". In: *Engineering Optimization* 36.4 (2004), pp. 419–438.

[69] Adel Younis and Zuomin Dong. "Metamodelling and search using space exploration and unimodal region elimination for design optimization". In: *Engineering Optimization* 42.6 (2010), pp. 517–533.

[70] G. Gary Wang, Zuomin Dong, and Peter Aitchison. "Adaptive response surface method - A global optimization scheme for approximation-based design problems". In: *Engineering Optimization* 33.6 (2001), pp. 707–733.

[71] S. Shan and G. G. Wang. "Space exploration and global optimization for computationally intensive design problems: A rough set based approach". In: *Structural and Multidisciplinary Optimization* 28.6 (2004), pp. 427–441.

[72] Qingfeng Zhuge et al. "Design optimization and space minimization considering timing and code size via retiming and unfolding". In: *Microprocessors and Microsystems* 30.4 (2006), pp. 173–183.

[73] Vinicius Veloso De Melo et al. "Improving global numerical optimization using a search-space reduction algorithm". In: *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference* January (2007), pp. 1195–1202.

[74] Huachao Dong et al. "SCGOSR : Surrogate-based constrained global optimization using space reduction". In: *Applied Soft Computing Journal* 65 (2018), pp. 462–477.

[75] Piotr Breitkopf et al. "Moving least squares response surface approximation: Formulation and metal forming applications". In: *Computers and Structures* 83.17-18 (2005), pp. 1411–1428.

[76] M. Oudjene et al. "Shape optimization of clinching tools using the response surface methodology with Moving Least-Square approximation". In: *Journal of Materials Processing Technology* 209.1 (2009), pp. 289–296.

[77] Vassili V. Toropov. "Simulation approach to structural optimization". In: *Structural Optimization* 1.1 (Mar. 1989), pp. 37–46.

[78] Andrey Polynkin and Vassili V. Toropov. "Mid-range metamodel assembly building based on linear regression for large scale optimization problems". In: *Structural and Multidisciplinary Optimization* 45.4 (Apr. 2012), pp. 515–527.

[79] Dianzi Liu and Vassili V. Toropov. "Implementation of Discrete Capability into the Enhanced Multipoint Approximation Method for Solving Mixed Integer-Continuous Optimization Problems". In: *International Journal of Computational Methods in Engineering Science and Mechanics* 17.1 (2016).

[80] Stefano Caloni, Shahrokh Shahpar, and Vassili V. Toropov. "Multi-Disciplinary Design Optimisation of the Cooled Squealer Tip for High Pressure Turbines". In: *Aerospace* 5.116 (2018).

[81] Juliane Müller and Joshua D. Woodbury. "GOSAC: global optimization with surrogate approximation of constraints". In: *Journal of Global Optimization* 69.1 (2017), pp. 117–136.

[82] Alexander I.J. Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. Wiley, 2008.

[83] Zbigniew Michalewicz and Marc Schoenauer. "Evolutionary Algorithms for Constrained Parameter Optimization Problems". In: *Evolutionary Computation* 4.1 (Mar. 1996), pp. 1–32.

[84] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Vol. 33. 5/6. Springer, June 2004, pp. 1064–1065.

[85] Oliver Kramer. "A Review of Constraint-Handling Techniques for Evolution Strategies". In: *Applied Computational Intelligence and Soft Computing* 2010.1 (2010), pp. 1–11.

[86] Efrén Mezura-montes and Carlos A. Coello Coello. "Constraint-handling in nature-inspired numerical optimization : Past , present and future". In: *Swarm and Evolutionary Computation* 1.4 (2011), pp. 173–194.

[87] Licheng Jiao et al. "A novel selection evolutionary strategy for constrained optimization". In: *Information Sciences* 239 (2013), pp. 122–141.

[88] Sergey Gorshkov. *Advances in Swarm and Computational Intelligence*. Vol. 9142. 2015, pp. 185–193.

[89] Carlos A. Coello Coello. "Constraint-handling techniques used with evolutionary algorithms". In: *GECCO 2018 Companion - Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion* (2018), pp. 773–799.

[90] Thomas Philip Runarsson. "Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models". In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Xin Yao et al. Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2004, pp. 401–410.

[91] Liang Shi and Khaled Rasheed. "ASAGA: An adaptive surrogate-assisted genetic algorithm". In: *GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008* (2008), pp. 1049–1056.

[92] M. J.D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Ed. by Susana Gomez and JP. Hennart. Dordrecht: Springer, Dordrecht, 1994, pp. 51–67.

[93] Jing Liang et al. "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization". In: *Nangyang Technological University, Singapore, Tech. Rep* 41 (2006).

[94] Marc Schoenauer and Zbigniew Michalewicz. "Evolutionary computation at the edge of feasibility". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1141 (1996), pp. 245–254.

[95] Kalyanmoy Deb. "An efficient constraint handling method for genetic algorithms". In: *Computer Methods in Applied Mechanics and Engineering* 186.2 (2000), pp. 311–338.

[96] Zbigniew Michalewicz and David B. Fogel. "How to Solve It: Modern Heuristics". In: *How to Solve It: Modern Heuristics* 49.0 (2004), p. 6221.

[97] Efrén Mezura-Montes and Carlos A. Coello Coello. "A simple multimembered evolution strategy to solve constrained optimization problems". In: *IEEE Transactions on Evolutionary Computation* 9.1 (2005), pp. 1–17.

[98] Piya Chootinan and Anthony Chen. "Constraint handling in genetic algorithms using a gradient-based repair method". In: *Computers and Operations Research* 33.8 (2006), pp. 2263–2281.

[99] Oliver Kramer, André Barthelmes, and Günter Rudolph. *Surrogate Constraint Functions for CMA Evolution Strategies*. 2009, pp. 169–176.

[100] Erwie Zahara and Yi Tung Kao. "Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems". In: *Expert Systems with Applications* 36 (2009), pp. 3880–3886.

[101] Patrick Koch et al. "A New Repair Method For Constrained Optimization". In: *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*. GECCO '15 September. New York, New York, USA: ACM Press, 2015, pp. 273–280.

[102] Carlos A. Coello Coello, Efrén Mezura Montes, and Efrén Mezura Montes. "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection". In: *Advanced Engineering Informatics* 16.3 (July 2002), pp. 193–203.

[103] Christian Setzkorn, Azzam F.G. Taktak, and Bertil E. Damato. "On the use of multi-objective evolutionary algorithms for survival analysis". In: *BioSystems* 87.1 (2007), pp. 31–48.

[104] Christodoulos Floudas and Panos Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer, 1990.

[105] Raziyeh Farmani and Jonathan A. Wright. "Self-Adaptive Fitness Formulation for Constrained Optimization". In: *IEEE Transactions on Evolutionary Computation* 7.5 (2003), pp. 445–455.

[106] Arturo Hernández Aguirre et al. *COPSO : Constrained Optimization via PSO algorithm*. Tech. rep. Center for Research in Mathematics. CIMAT, 2007.

[107] Thomas Philip Runarsson and Xin Yao. "Search biases in constrained evolutionary optimization". In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 35.2 (2005), pp. 233–243.

[108] Yong Wang et al. "An adaptive tradeoff model for constrained evolutionary optimization". In: *IEEE Transactions on Evolutionary Computation* 12.1 (2008), pp. 80–92.

[109] Rammohan Mallipeddi and Ponnuthurai N. Suganthan. "Ensemble of constraint handling techniques". In: *IEEE Transactions on Evolutionary Computation* 14.4 (2010), pp. 561–579.

[110] Yong Wang et al. "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37.3 (2007), pp. 560–575.

[111] Tetsuyuki Takahama and Setsuko Sakai. "Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations". In: *IEEE Transactions on Evolutionary Computation* 9.5 (2005), pp. 437–451.

[112] Wen Long et al. "An effective hybrid cuckoo search algorithm for constrained global optimization". In: *Neural Computing and Applications* 25.3-4 (2014), pp. 911–926.

[113] Ali Husseinzadeh Kashan. "An efficient algorithm for constrained global optimization and application to mechanical engineering design : League championship algorithm (LCA)". In: *Computer-Aided Design* 43.12 (2011), pp. 1769–1792.

[114] Rommel G. Regis. "Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions". In: *Computers and Operation Research* 38.5 (2011), pp. 837–853.

[115] Rommel G. Regis. "Constrained Optimization by Radial Basis Function Interpolation for High-Dimensional Expensive Black-Box Problems with Infeasible Initial Points". In: *Engineering Optimization* 46.2 (2014), pp. 218–243.

[116] Samineh Bagheri et al. "Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets". In: *Applied Soft Computing Journal* 61 (2017), pp. 377–393.

[117] Yaohui Li et al. "A Kriging-based constrained global optimization algorithm for expensive black-box functions with infeasible initial points". In: *Journal of Global Optimization* 67.1-2 (2017), pp. 343–366.

[118] Haitao Liu et al. "Constrained global optimization via a DIRECT-type constraint-handling technique and an adaptive metamodeling strategy". In: *Structural and Multidisciplinary Optimization* 55.1 (2016), pp. 155–177.

[119] Donald R. Jones. *Large-Scale Multi-Disciplinary Mass Optimization in the Auto Industry*. Tech. rep. 2008.

[120] Luc Pronzato. "Minimax and maximin space-filling designs: some properties and methods for construction". In: *Journal de la Société Française de Statistique* 158.1 (2017), pp. 7–36.

[121] Laura P Swiler and Gregory D Wyss. *A User's Guide to Sandia's Latin Hypercube Sampling Software: LHS UNIX Library/Standalone Version*. Tech. rep. July. Sandia National Laboratories, 2004.

[122] Jayant R. Kalagnanam and Urmila M. Diwekar. "An efficient sampling technique for off-line quality control". In: *Technometrics* 39.3 (1997), pp. 308–319.

[123] Bruce Jay Collings and Harald Niederreiter. "Random Number Generation and Quasi-Monte Carlo Methods." In: *Journal of the American Statistical Association* 88.422 (1993), p. 699.

[124] Tien-Tsin Wong, Wai-Shing Luk, and Pheng-Ann Heng. "Sampling with Hammersley and Halton Points". In: *Journal of Graphics Tools* 2.2 (1997), pp. 9–24.

[125] Georges Matheron. "Principles of geostatistics". In: *Economic Geology* 58.8 (1963), pp. 1246–1266.

[126] Jack P.C. Kleijnen. "Kriging metamodeling in simulation: A review". In: *European Journal of Operational Research* 192.3 (2009), pp. 707–716.

[127] Søren N Lophaven, Hans Bruun Nielsen, and Jacob Søndergaard. *DACE - A Matlab Kriging Toolbox*. Tech. rep. Denmark: Technical University of Denmark, 2002.

[128] Alexander I.J. Forrester and Andy J. Keane. "Recent advances in surrogate-based optimization". In: *Progress in Aerospace Sciences* 45.1-3 (2009), pp. 50–79.

[129] R. L. Hardy. "Theory and applications of the multiquadric-biharmonic method". In: *Computers and Mathematics with Applications* 19.8-9 (1990), pp. 163–208.

[130] G Venter. "Review of optimization techniques". In: *Encyclopedia of Aerospace Engineering* July (2010), pp. 1–12.

[131] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680.

[132] Zhi-Hui Zhan et al. "Adaptive Particle Swarm Optimization". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.6 (2009), pp. 1362–1381.

[133] Riccardo Poli, James Kennedy, and Tim Blackwell. "Particle swarm optimization". In: *Swarm Intelligence* 1 (2007), pp. 33–57.

[134] Maurice Clerc and James Kennedy. "The particle swarm-explosion, stability, and convergence in a multidimensional complex space". In: *IEEE Transactions on Evolutionary Computation* 6.1 (2002), pp. 58–73.

[135] Vassili V. Toropov, A. A. Filatov, and A. A. Polynkin. "Multiparameter structural optimization using FEM and multipoint explicit approximations". In: *Structural Optimization* 6.1 (Mar. 1993), pp. 7–14.

[136] George E P Box and Norman R Draper. *Empirical Model-Building and Response Surfaces*. 1987, p. 669.

[137] Fred Van Keulen and Vassili V. Toropov. "New Developments in Structural Optimization Using Adaptive Mesh Refinement and Multipoint Approximations". In: *Engineering Optimization* 29.1-4 (Oct. 1997), pp. 217–234.

[138] Guido van Rossum. *Foreword for "Programming Python" (1st ed.)* 2014. URL: `https://www.python.org/doc/essays/foreword/`.

[139] Stack Overflow. *Stack Overflow Annual Developer Survey*. 2019. URL: `https://insights.stackoverflow.com/survey/2019`.

[140] TIOBE. *TIOBE Index*. 2020. URL: `https://www.tiobe.com/tiobe-index/`.

[141] Python Software Foundation. *The Python Standard Library*. 2020. URL: `https://docs.python.org/3/library/index.html`.

[142] Python Software Foundation. *The Python Package Index*. 2020. URL: `https://pypi.org/`.

[143] Travis Oliphant and Jarrod k. Millma. *A guide to NumPy*. 2006.

[144] Stéfan Van Der Walt, S. Chris Colbert, and Gaël Varoquaux. "The NumPy array: A structure for efficient numerical computation". In: *Computing in Science and Engineering* (2011).

[145] Fabian Pedregosa et al. "Scikit-learn". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[146] Pauli Virtanen et al. "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python". In: *arXiv e-prints* (2019), pp. 1–22.

[147] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. "Numba: A LLVM-based Python JIT Compiler". In: *LLVM '15 Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. Austin, Texas: ACM, 2015.

[148] The Spyder Website Contributors. *SPYDER: The Scientific Python Development Envionment*. 2020. URL: `https://www.spyder-ide.org/`.

[149] Computational Mathematics Group. "Harwell Subroutine Library". In: ().

[150] E. Sandgren. “Nonlinear integer and discrete programming in mechnical design optimization”. In: *Journal of Mechanisms, Transmissions, and Automation in Design* 112.2 (1990), pp. 223–229.

[151] Takashi Okamoto and Hironori Hirata. “Constrained optimization using the quasi-chaotic optimization method with the exact penalty function and the sequential quadratic programming”. In: *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2011, pp. 1765–1770.

[152] Wenyin Gong, Zhihua Cai, and Dingwen Liang. “Engineering optimization by means of an improved constrained differential evolution”. In: *Computer Methods in Applied Mechanics and Engineering* 268 (2014), pp. 884–904.

[153] R. de Paula Garcia, B. S. L. P. de Lima, and A. C. de Castro Lemonge. “A Surrogate Assisted Differential Evolution to Solve Constrained Optimization Problems”. In: *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. Arequipa, Peru: IEEE, 2017, pp. 1–6.

[154] Rommel G. Regis and Stefan M. Wild. “CONORBIT: constrained optimization by radial basis function interpolation in trust regions”. In: *Optimization Methods and Software* 32.3 (2017), pp. 552–580.

[155] Stefan M. Wild, Rommel G. Regis, and Christine A Shoemaker. “ORBIT: Optimization by Radial Basis Function Interpolation in Trust-Regions”. In: *SIAM J. SCI. COMPUT.* 30.6 (2008), pp. 3197–3219.

[156] Stefan M. Wild and Christine Shoemaker. “Global convergence of radial basis function trust region derivative-free algorithms”. In: *SIAM Journal on Optimization* 21.3 (2011), pp. 761–781.

[157] Kalyanmoy Deb and Soumil Srivastava. “A genetic algorithm based augmented Lagrangian method for constrained optimization”. In: *Computational Optimization and Applications* 53.3 (Dec. 2012), pp. 869–902.

[158] Mark A. Abramson and Charles Audet. “Convergence of mesh adaptive direct search to second-order stationary points”. In: *SIAM Journal on Optimization* 17.2 (2006), pp. 606–619.

[159] Charles Audet and J. E. Dennis. “Mesh adaptive direct search algorithms for constrained optimization”. In: *SIAM Journal on Optimization* 17.2 (2007), pp. 188–217.

[160] Rommel G. Regis and Christine A. Shoemaker. "A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions". In: *INFORMS Journal on Computing* 19.4 (Nov. 2007), pp. 497–509.

[161] A. Forrester and Donald R. Jones. "Enhancements to the expected improvement criterion". In: *Presented at the 20th international symposium on mathematical programming (ISMP)*. Chicago, 2009.

[162] N Quttineh and Kenneth Holmström. "Implementation of a one-stage EGO algorithm". In: *Presented at the 20th international symposium on mathematical programming (ISMP)*. Chicago, 2009.

[163] Rommel G. Regis. "Trust Regions in Surrogate-Assisted Evolutionary Programming for Constrained Expensive Black-Box Optimization". In: *Evolutionary Constrained Optimization*. Ed. by Rituparna Datta and Kalyanmoy Deb. New Delhi: Springer India, 2015, pp. 51–94.

[164] Ami Marowka. "Python accelerators for high-performance computing". In: *Journal of Supercomputing* 74.4 (2018), pp. 1449–1460.

[165] Python Software Foundation. *Multiprocessing*. 2020. URL: `https://docs.python.org/3.8/library/multiprocessing.html`.

[166] Rommel G. Regis. "Evolutionary Programming for High-Dimensional Constrained Expensive Black-Box Optimization Using Radial Basis Functions". In: *IEEE Transactions on Evolutionary Computation* 18.3 (2014), pp. 326–347.

[167] Vassili V. Toropov. "Modelling and Approximation Strategies in Optimization: Global and Mid-Range Approximations, Response Surface Methods, Genetic Programming, Low/High Fidelity Models". In: *Emerging Methods for Multidisciplinary Optimization* 1997 (2001), pp. 205–256.

[168] Najeh Ben Guedria. "Improved accelerated PSO algorithm for mechanical engineering optimization problems". In: *Applied Soft Computing Journal* 40 (2016), pp. 455–467.

Appendices

A

Benchmark library

A.1 Engineering benchmark problems

Four well-known engineering design optimization problems are included in the library for benchmark testing. There are two continuous constrained optimization problems: the welded beam design problem (WBD) [102] and the tension/compression spring design problem [150]. The rest two problems , i.e., the pressure vessel design problem [150] and the speed reducer design problem [104] are mixed continuous/discrete variable optimization problems.

A.1.1 Welded Beam Design

As shown in Fig. A.1, the beam is welded to a rigid support and is designed for the minimum cost, considering constraints on shear stress (τ), bending stress (σ), buckling load (p_c), and end deflection (δ). The design variables comprise the thickness of the weld (x_1), the length of the welded joint (x_2), the width of the beam (x_3) and the thickness of the beam (x_4). The problem can be formulated mathematically as Equation A.1.1.

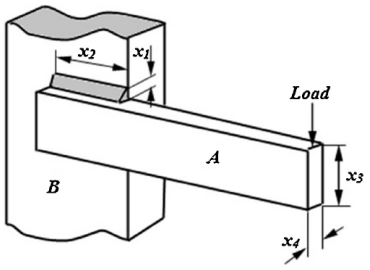


Figure A.1: Schematic view of the welded beam structure from [168]

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \\
s.t. \quad & g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max} \leq 0 \\
& g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max} \leq 0 \\
& g_3(\boldsymbol{x}) = x_1 - x_4 \leq 0 \\
& g_4(\boldsymbol{x}) = [0.10471x_1^2 + 0.04811x_3x_4(14 + x_2)] - 5 \leq 0 \\
& g_5(\boldsymbol{x}) = 0.125 - x_1 \leq 0 \\
& g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{max} \leq 0 \\
& g_7(\boldsymbol{x}) = p - p_c(\boldsymbol{x}) \leq 0 \\
where \quad & P = 6000\,lb,\ L = 14\,in,\ E = 30 \times 10^6\,psi,\ G = 12 \times 10^6\,psi, \\
& \tau_{max} = 13600\,psi,\ \sigma_{max} = 30000\,psi,\ \delta_{max} = 0.25\,in \\
& \tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right) \\
& R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
& \tau\,(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\
& J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \\
& \sigma\,(x) = \frac{6PL}{x_4x_3^2}, \delta\,(x) = \frac{4PL^3}{Ex_3^3x_4} \\
& p_c\,(x) = \frac{4.013\sqrt{E\left(\frac{x_3^2x_4^6}{36}\right)}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \\
& 0.1 \leq x_1 \leq 2,\ 0.1 \leq x_2 \leq 10,\ 0.1 \leq x_3 \leq 10,\ 0.1 \leq x_4 \leq 2
\end{aligned}
\tag{A.1.1}
$$

A.1.2 Tension/compression spring design

As shown in Figure A.2, the design variables include the wire diameter $d\,(x_1)$, the mean coil diameter $D\,(x_2)$, and the number of active coils $N\,(x_3)$. The design objective is to minimize the weight of the spring subject to constraints on the minimum deflection g_1, shear stress g_2, surge frequency g_3 and the limits on the outside diameter g_4. The mathematical description of this problem is given as follows:

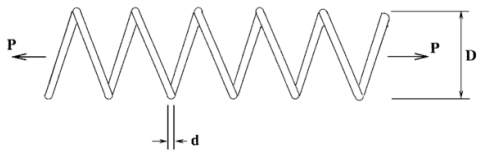


Figure A.2: Schematic view of the spring structure

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = x_1^2 x_2 (x_3 + 2) \\
s.t. \quad & g_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\
& g_2(x) = \frac{4x_2^2 - x_2 x_1}{12566 \left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\
& g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\
& g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \\
where \quad & 0.05 \leq x_1 \leq 1;\ 0.25 \leq x_2 \leq 1.3;\ 2 \leq x_3 \leq 15.
\end{aligned}
\tag{A.1.2}
$$

A.1.3 Pressure vessel design

Figure A.3 shows a cylindrical pressure vessel capped at both ends by hemispherical heads. According to the American society of mechanical engineers (ASME) boiler and pressure vessel code, this vessel is designed for a working pressure of $3000\,psi$ and a minimum volume of $750\,ft^3$. The objective is to minimize the total cost which involves a welding cost, a material cost and a forming cost. The variables include the thickness of shell (x_1), the thickness of the head (x_2), the inner radius (x_3), and the length of the cylindrical section of the vessel (x_4). Among them, x_3 and x_4 are continuous variables but the thickness x_1 and x_2 can only take integer multiples of $0.0625\,inch$. The mathematical expression of this problem is given in Equation A.3.

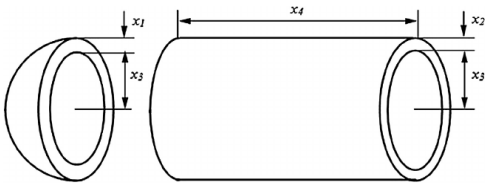


Figure A.3: Schematic view of the pressure vessel

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_1x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
s.t. \quad & g_1(x) - x_1 + 0.0193x_3 \leq 0 \\
& g_2(x) = -x_2 + 0.00954x_3 \leq 0 \\
& g_3(x) = -\pi x_3^2x_4^2 - \frac{4}{3}\pi x_3^3 + 129600 \leq 0 \\
& g_4(x) = x_4 - 240 \leq 0 \\
where \quad & 1 \times 0.0625 \leq x_1, x_2 \leq 99 \times 0.0625 \\
& 10.0 \leq x_3, x_4 \leq 200.0
\end{aligned}
\tag{A.1.3}
$$

A.1.4 Speed reducer design

A speed reducer as shown in Figure A.4 is part of the gear box of mechanical system. The total weight of the speed reducer is to be minimized subject to the nine constraints which include the limits on the bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. The design variables are the face width (x_1), the module of the teeth (x_2), the number of teeth on pinion (x_3), the length of the first shaft between bearings (x_4), the length of the second shaft between bearings (x_5), the diameter of the first and the second shaft (x_6 and x_7). Among them, x_3 is integer and the rest are continuous. The mathematical formulation can be summarized in Equation A.1.4.

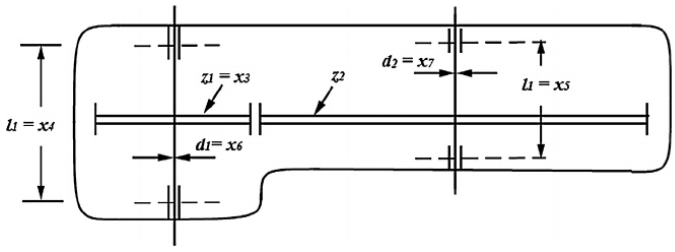


Figure A.4: Schematic view of the speed reducer

$$
\begin{aligned}
min \quad & f(\boldsymbol{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
& - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \\
s.t. \quad & g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0 \\
& g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0 \\
& g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0 \\
& g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0 \\
& g_5(x) = \frac{1.0}{110x_6^3}\sqrt{(\frac{745.0x_4}{x_2x_3})^2 + 16.9 \times 10^6} - 1 \leq 0 \\
& g_6(x) = \frac{1.0}{110x_6^3}\sqrt{(\frac{745.0x_4}{x_2x_3})^2 + 16.9 \times 10^6} - 1 \leq 0 \\
& g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0 \\
& g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0 \\
& g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0 \\
& g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0 \\
& g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0 \\
where \quad & 2.6 \leq x_1 \leq 3.6,\ 0.7 \leq x_2 \leq 0.8,\ 17 \leq x_3 \leq 28 \\
& 7.3 \leq x_4 \leq 8.3,\ 7.8 \leq x_5 \leq 8.3,\ 2.9 \leq x_6 \leq 3.9 \\
& 5.0 \leq x_7 \leq 5.5
\end{aligned}
\tag{A.1.4}
$$

A.2 G-problems

For details about G-problems, readers are referred to [93].

A.3 MOPTA08

The fortran files about the MOPTA08 problem are given in https://www.miguelanjos.com/jones-benchmark.

B

Convergence plots

In this chapter, the convergence graph of each benchmark problem optimized by different methods are given below. The meaning of the plots has been described in Section 3.6.3. For ease of understanding, the definition is repeated here. The graph shows the median error between the objective value of the sub-optimal solution in k_{th} iteration $f(\boldsymbol{x}^k)$ and the global known optimum $f(\boldsymbol{x}^*)$ versus the median NFEs at k_{th} iteration in all trials. The error bars mark the 25% and 75% quartile. As shown in the legend plot (Figure B.1) The red point means that in median trials, the method can not find a feasible sub-optimal solution at the median NFEs. And if the method can obtain a feasible solution at this iteration in median runs, the point is marked as a yellow triangle. In this case, if the solution also satisfies the success condition ($f(\boldsymbol{x}^k) - f(\boldsymbol{x}^*) \leq 1e-4$), it will be denoted as a green star instead.

B.1 IATRO convergence graphs

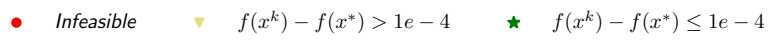
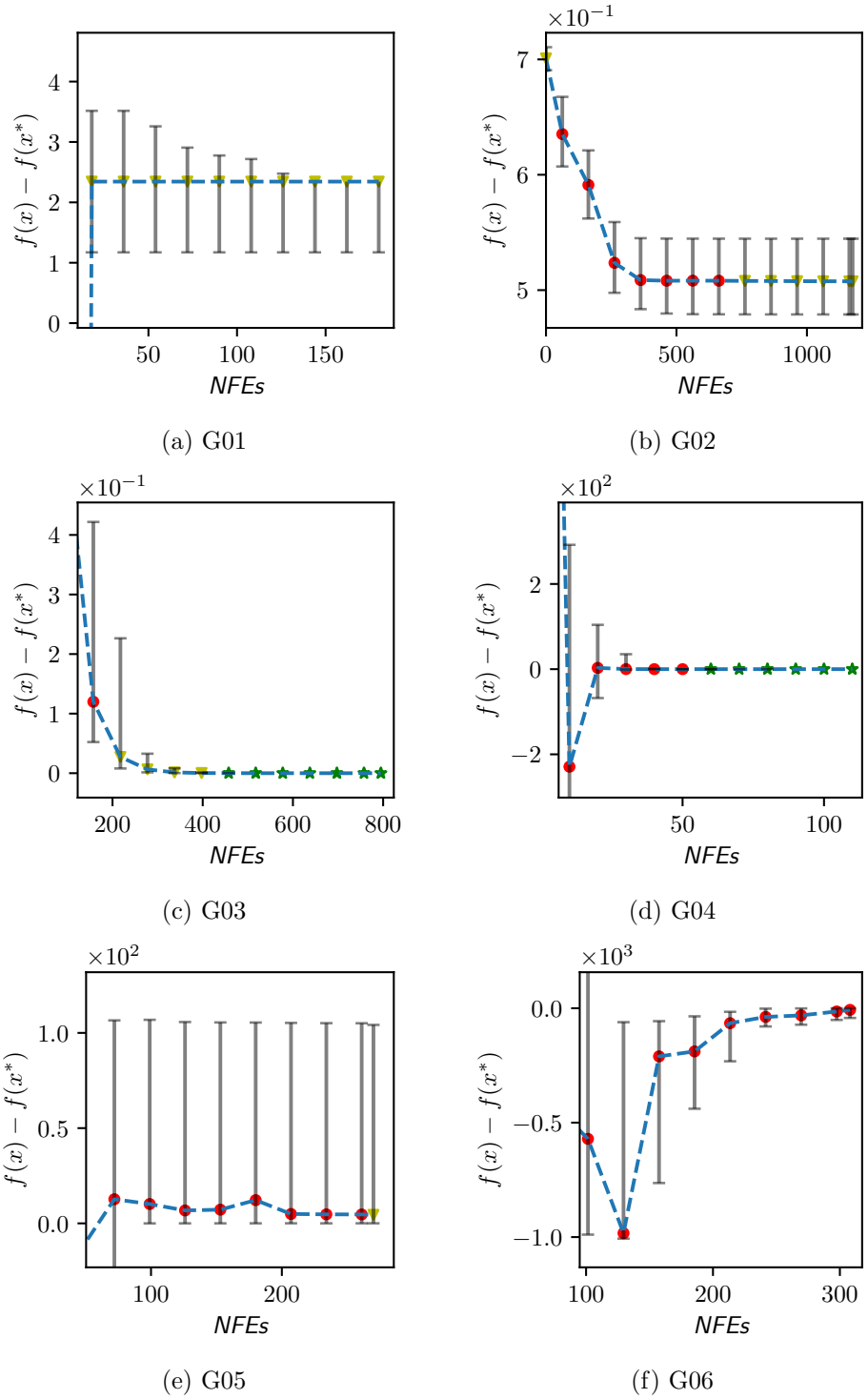


Figure B.1: The legend used in convergence graphs

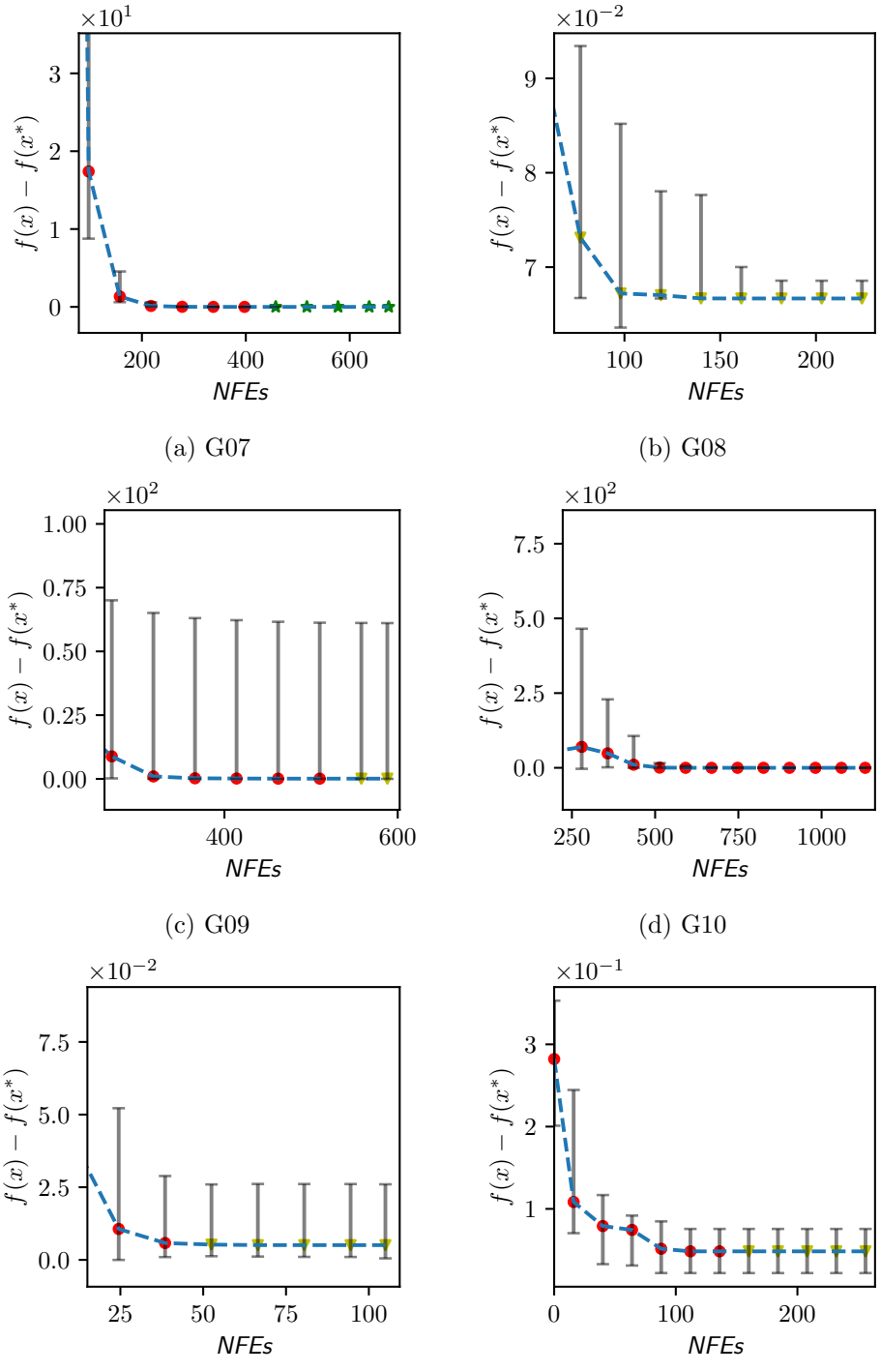


(a) G01 (b) G02

(c) G03 (d) G04

(e) G05 (f) G06

Figure B.2: IATRO optimization process for G01-G06

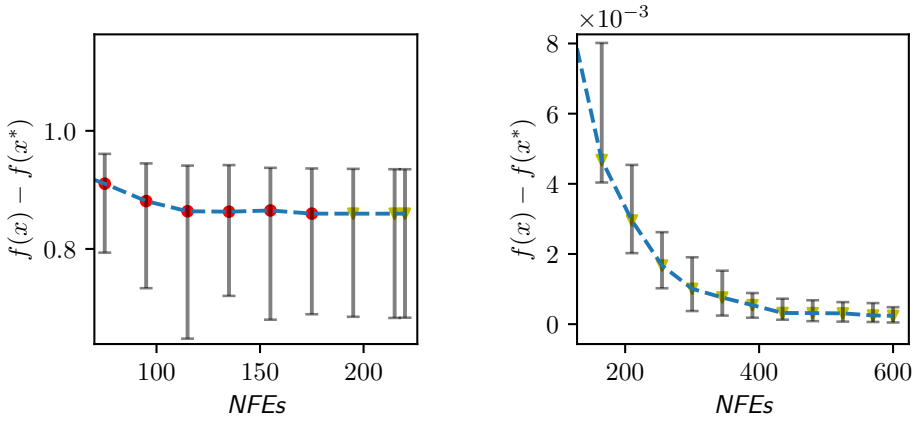


(a) G07 (b) G08

(c) G09 (d) G10

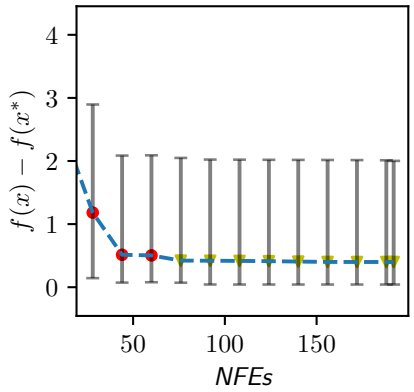
(e) G11 (f) G12

Figure B.3: IATRO optimization process for G07-G12

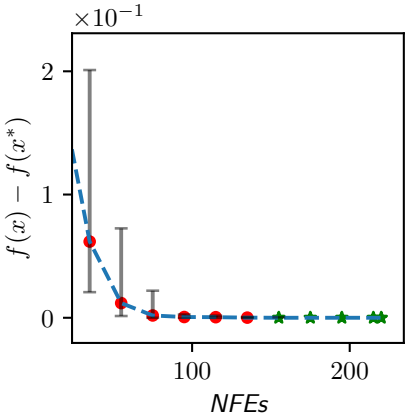


(a) G13

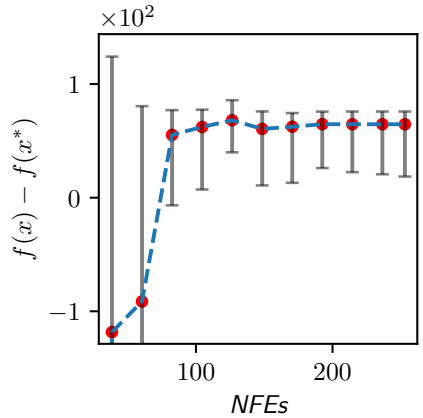
(b) G14



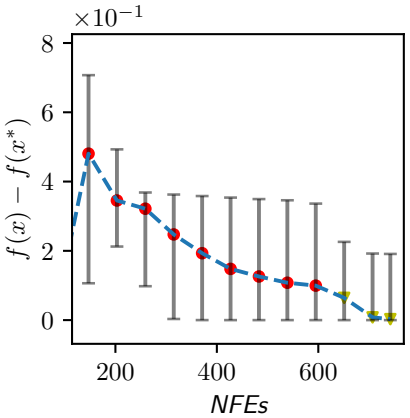
(c) G15



(d) G16

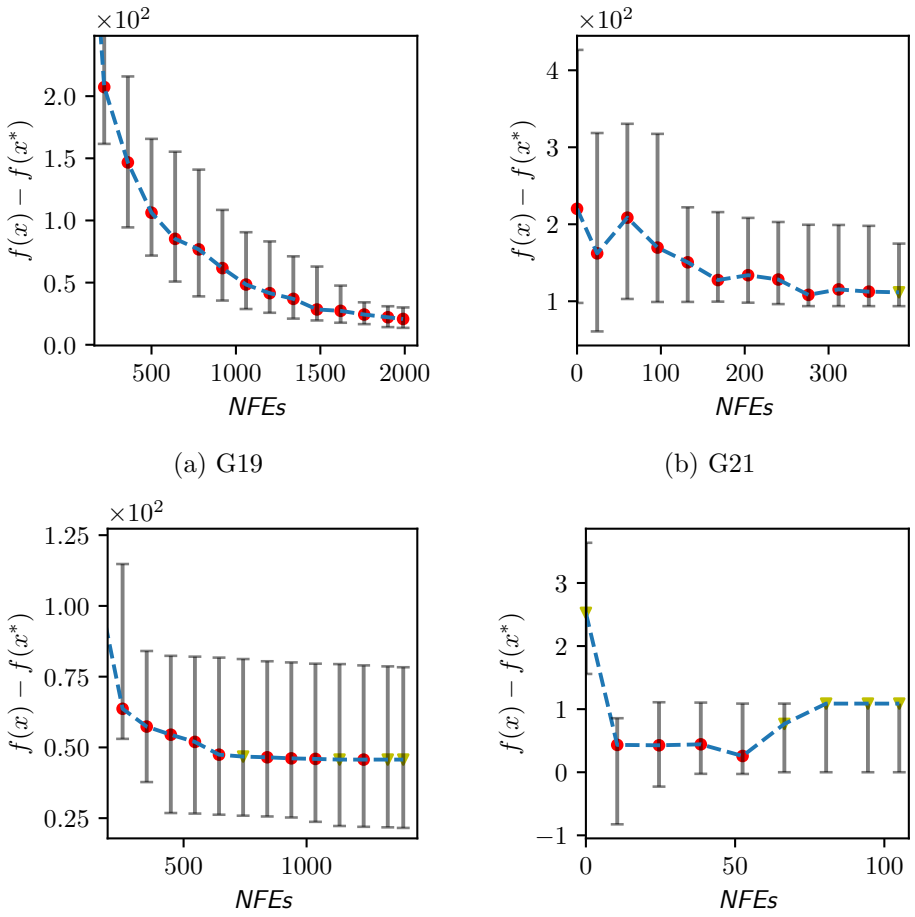


(e) G17



(f) G18

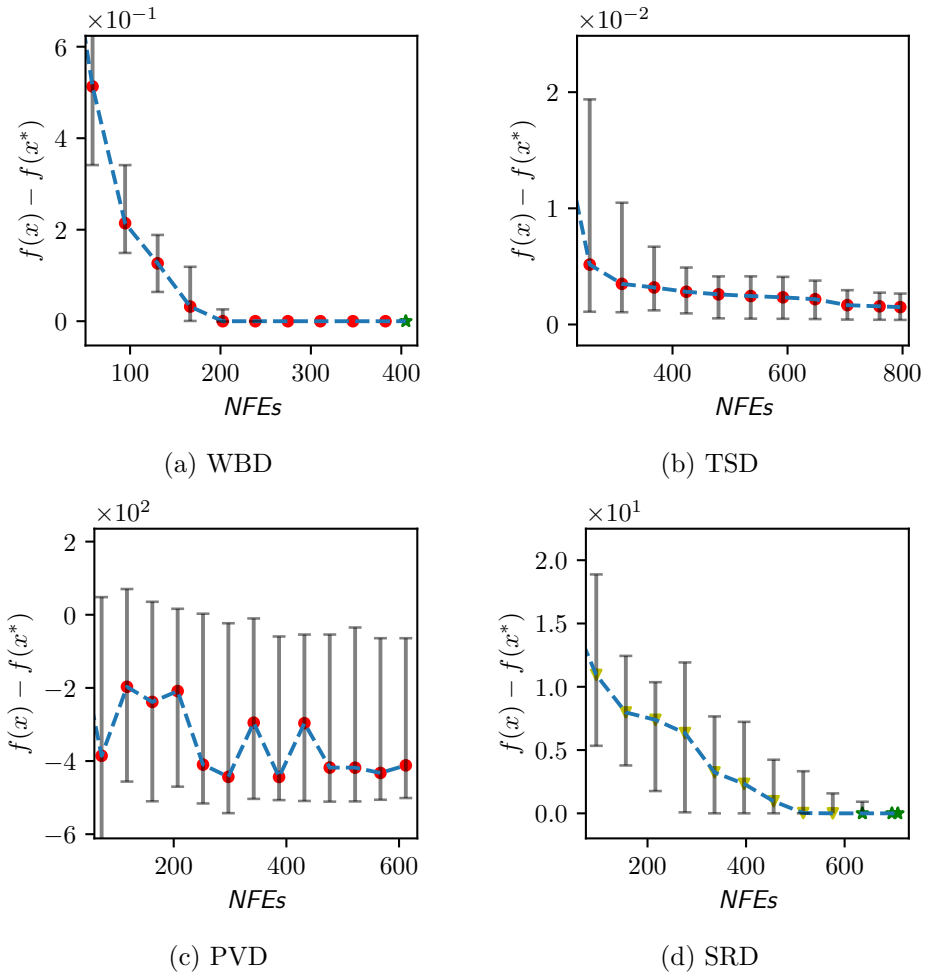
Figure B.4: IATRO optimization process for G13-G18



(a) G19 (b) G21

(c) G23 (d) G24

Figure B.5: IATRO optimization process for G19-G24



(a) WBD (b) TSD

(c) PVD (d) SRD

Figure B.6: IATRO optimization process for engineering design problems

C

Optimization statistics

In this chapter, the optimization statistics of an algorithm on the benchmark problems are given in details, while in the main text maybe only the significant parts of the statistics are used to make comparisons.

C.1 IE (IATRO–ESS)

Table C.1: Statistical results of the optimal objective function values obtained by IATRO–ESS ($\eta_{eco} = 50\%$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-11.2813	-13.1638	-13.8281	9.5745e-01
G02	-0.8036	-0.4376	-0.2156	-0.2890	-0.2730	6.0195e-02
G03	-1.0005	-1.0005	-0.0000	-0.9581	-1.0005	1.9592e-01
G04	-30665.5387	-30665.5397	-30665.5348	-30665.5388	-30665.5387	8.6991e-04
G05	5126.4981	5126.4987	5949.6864	5231.9149	5137.9311	2.2405e+02
G06	-6961.8139	-6961.8138	-4147.2655	-6612.6497	-6952.2640	6.9119e+02
G07	24.3062	24.3062	24.3064	24.3062	24.3062	3.8562e-05
G08	-0.0958	-0.0958	-0.0167	-0.0548	-0.0291	3.0732e-02
G09	680.6301	680.6301	815.9856	697.0905	680.6301	3.8454e+01
G10	7049.2480	7049.2500	7508.8089	7134.6669	7050.0633	1.5909e+02
G11	0.7500	0.7500	0.9408	0.8049	0.7811	5.9569e-02
G12	-1.0000	-1.0000	-0.8781	-0.9528	-0.9606	4.0602e-02
G13	0.0539	0.2071	1.3158	0.8421	0.9731	2.9405e-01
G14	-47.7611	-47.7611	-47.7547	-47.7599	-47.7603	1.4767e-03
G15	961.7152	961.7152	969.8983	963.2743	962.1370	2.1522e+00
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	1.5443e-08
G17	8876.9807	8862.9014	8878.6607	8870.7810	8870.7810	7.8796e+00
G18	-0.8660	-0.8660	-0.4992	-0.7187	-0.8636	1.6330e-01
G19	32.6556	35.8172	244.7116	81.7079	69.5000	4.7990e+01
G21	193.7869	238.4943	518.3504	324.2144	292.0188	7.8826e+01
G23	-400.0000	-400.0000	-24.6838	-361.3975	-376.6007	7.2867e+01
G24	-5.5080	-5.5080	-4.0537	-5.2227	-5.5080	4.7741e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.1107e-10
SPD	0.0127	0.0127	0.0217	0.0138	0.0132	1.7638e-03
PVD	6059.7143	6082.5117	9261.4445	6534.3014	6378.9278	6.2348e+02
SRD	2994.4710	2994.4696	3000.4098	2994.7101	2994.4707	1.1635e+00

Table C.2: Convergence statistics of IATRO–ESS ($\eta_{eco} = 50\%$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	188	120	0.64	100	4	4710	120
G02	1187	1181	0.99	100	0	-	1181
G03	799	725	0.91	100	92	869	725
G04	103	71	0.69	100	96	107	71
G05	200	186	0.93	96	0	-	194
G06	103	65	0.63	80	4	3209	81
G07	695	689	0.99	100	96	724	689
G08	165	126	0.76	100	24	689	126
G09	592	587	0.99	92	48	1340	638
G10	827	462	0.56	40	0	-	1156
G11	115	99	0.86	100	12	959	99
G12	183	119	0.65	100	4	4565	119
G13	194	179	0.92	92	0	-	195
G14	618	615	1.00	100	12	5148	615
G15	134	126	0.94	80	4	4200	157
G16	155	135	0.87	100	100	155	135
G17	217	125	0.58	8	4	67812	1562
G18	760	656	0.86	100	44	1726	656
G19	1889	1812	0.96	100	0	-	1812
G21	256	252	0.98	100	0	-	252
G23	884	862	0.98	96	12	7670	898
G24	72	50	0.69	100	72	100	50
WBD	195	190	0.97	100	100	195	190
SPD	501	482	0.96	100	16	3129	482
PVD	433	150	0.35	100	0	-	150
SRD	507	428	0.84	100	88	576	428

C.2 IEM (IATRO–ESS–MTRS)

Table C.3: Statistical results of the optimal objective function values obtained by IATRO–ESS–MTRS ($\eta_{eco} = 50\%$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-11.2813	-13.3569	-13.8281	1.0986e+00
G02	-0.8036	-0.4805	-0.2293	-0.3434	-0.3404	7.1015e-02
G03	-1.0005	-1.0005	-0.9937	-1.0002	-1.0005	1.3280e-03
G04	-30665.5387	-30665.5398	-30665.5387	-30665.5390	-30665.5388	3.0090e-04
G05	5126.4981	5126.4981	5623.2905	5222.5940	5156.2180	1.5712e+02
G06	-6961.8139	-6961.8139	-5769.6011	-6812.1923	-6958.8701	3.2794e+02
G07	24.3062	24.3062	24.3062	24.3062	24.3062	4.6964e-07
G08	-0.0958	-0.0958	-0.0004	-0.0617	-0.0560	3.2749e-02
G09	680.6301	680.6301	1118.3159	698.8670	680.6301	8.7461e+01
G10	7049.2480	7049.2483	7805.5303	7132.6692	7049.6578	1.9119e+02
G11	0.7500	0.7500	0.9677	0.7854	0.7527	6.3679e-02
G12	-1.0000	-1.0000	-0.8594	-0.9642	-0.9694	3.6308e-02
G13	0.0539	0.0628	8.8485	1.4971	0.8292	1.8590e+00
G14	-47.7611	-47.7611	-47.7611	-47.7611	-47.7611	3.0761e-07
G15	961.7152	961.7152	968.3955	962.8212	961.7191	1.9881e+00
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	4.4614e-08
G17	8876.9807	8936.9796	9210.5234	9036.6467	8962.4371	1.2339e+02
G18	-0.8660	-0.8660	-0.5000	-0.7461	-0.8660	1.3720e-01
G19	32.6556	32.6556	32.9900	32.6703	32.6556	6.5373e-02
G21	193.7869	251.3768	821.4910	378.8182	314.8038	1.5591e+02
G23	-400.0000	-400.0004	-73.6739	-365.1203	-385.5032	6.5476e+01
G24	-5.5080	-5.5080	-4.0537	-5.0583	-5.5080	6.0922e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.4667e-11
SPD	0.0127	0.0127	0.0127	0.0127	0.0127	3.4494e-09
PVD	6059.7143	6059.7314	11075.5648	6682.5129	6387.9524	1.1174e+03
SRD	2994.4710	2994.4696	2994.4710	2994.4705	2994.4707	4.1678e-04

Table C.4: Convergence statistics of IATRO–ESS–MTRS ($\eta_{eco} = 50\%$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	297	123	0.41	100	20	1485	123
G02	2496	2451	0.98	100	0	-	2451
G03	1448	1389	0.96	100	96	1508	1389
G04	136	93	0.68	100	100	136	93
G05	295	282	0.96	100	4	7373	282
G06	118	81	0.69	64	4	4629	127
G07	890	870	0.98	100	100	890	870
G08	142	113	0.80	100	44	322	113
G09	716	679	0.95	96	92	811	708
G10	801	582	0.73	84	0	-	693
G11	208	190	0.91	100	8	2606	190
G12	225	148	0.66	100	4	5636	148
G13	275	249	0.91	100	0	-	249
G14	719	706	0.98	100	100	719	706
G15	378	363	0.96	88	36	1192	412
G16	174	137	0.79	100	100	174	137
G17	327	253	0.77	12	0	-	2108
G18	932	824	0.88	100	52	1792	824
G19	1712	1650	0.96	100	80	2139	1650
G21	385	363	0.94	100	0	-	363
G23	845	794	0.94	100	44	1920	794
G24	70	54	0.78	100	64	109	54
WBD	187	183	0.98	100	100	187	183
SPD	340	277	0.82	100	100	340	277
PVD	397	204	0.51	100	0	-	204
SRD	332	254	0.77	100	100	332	254

C.3 IEMS (IATRO–ESS–MTRS–SANS)

Table C.5: Statistical results of the optimal objective function values obtained by IATRO–ESS–MTRS–SANS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-11.2813	-13.0700	-12.6563	1.1544e+00
G02	-0.8036	-0.5753	-0.2259	-0.3616	-0.3198	9.8264e-02
G03	-1.0005	-1.0005	-0.9984	-1.0004	-1.0005	4.4961e-04
G04	-30665.5387	-30665.5394	-30665.5387	-30665.5388	-30665.5387	1.8047e-04
G05	5126.4981	5126.4982	6106.7217	5411.4481	5311.9247	2.9577e+02
G06	-6961.8139	-6961.8148	-6961.8139	-6961.8143	-6961.8142	3.2655e-04
G07	24.3062	24.3062	24.3062	24.3062	24.3062	4.4525e-07
G08	-0.0958	-0.0958	-0.0273	-0.0615	-0.0510	2.9642e-02
G09	680.6301	680.6301	680.6301	680.6301	680.6301	1.5090e-06
G10	7049.2480	7049.2480	8057.3072	7151.3674	7049.6109	2.6029e+02
G11	0.7500	0.7500	0.9845	0.7881	0.7507	7.1797e-02
G12	-1.0000	-1.0000	-0.7363	-0.9553	-0.9911	6.7420e-02
G13	0.0539	0.0669	21.0549	1.7366	0.8599	3.9943e+00
G14	-47.7611	-47.7611	-47.7610	-47.7611	-47.7611	1.8773e-05
G15	961.7152	961.7152	970.6458	963.4000	961.8930	2.5458e+00
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	1.6436e-08
G17	8876.9807	8858.7901	9266.1209	8994.1785	8950.2147	1.1585e+02
G18	-0.8660	-0.8660	-0.5000	-0.7756	-0.8660	1.1926e-01
G19	32.6556	32.6556	32.6630	32.6562	32.6556	1.6800e-03
G21	193.7869	230.5979	757.2806	407.2879	340.7696	1.6124e+02
G23	-400.0000	-400.0000	-40.1192	-358.9579	-389.2441	9.1496e+01
G24	-5.5080	-5.5080	-4.0537	-4.9776	-5.5080	5.7746e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.5284e-11
SPD	0.0127	0.0127	0.0127	0.0127	0.0127	5.8806e-09
PVD	6059.7143	6059.7110	7143.4753	6255.9242	6179.6878	2.4677e+02
SRD	2994.4710	2994.4697	2994.4711	2994.4707	2994.4707	2.8055e-04

Table C.6: Convergence statistics of IATRO–ESS–MTRS–SANS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	308	134	0.43	100	12	2564	134
G02	2496	2472	0.99	100	0	-	2472
G03	1441	1404	0.97	100	92	1566	1404
G04	162	82	0.50	100	100	162	82
G05	189	174	0.92	100	0	-	174
G06	90	82	0.91	100	100	90	82
G07	855	833	0.97	100	100	855	833
G08	147	112	0.76	100	40	369	112
G09	744	733	0.98	100	100	744	733
G10	804	524	0.65	92	12	7279	570
G11	172	150	0.87	100	40	431	150
G12	220	140	0.64	100	4	5499	140
G13	271	253	0.93	100	0	-	253
G14	792	784	0.99	100	100	792	784
G15	180	139	0.77	92	8	2442	151
G16	178	145	0.82	100	100	178	145
G17	247	229	0.92	100	4	6184	229
G18	893	814	0.91	100	60	1488	814
G19	1737	1702	0.98	100	80	2171	1702
G21	391	350	0.89	100	0	-	350
G23	897	863	0.96	100	44	2038	863
G24	70	52	0.75	100	52	134	52
WBD	164	160	0.98	100	100	164	160
SPD	333	275	0.83	100	100	333	275
PVD	386	222	0.57	100	8	4824	222
SRD	326	256	0.78	100	100	326	256

C.4 RESM (RBFI–ESS–SANS–MTRS)

Table C.7: Statistical results of the optimal objective function values obtained by RBFI–ESS–SANS–MTRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-11.4844	-13.6406	-13.8281	9.7653e-01
G02	-0.8036	-0.7042	-0.2540	-0.3818	-0.3579	1.0785e-01
G03	-1.0005	-1.0005	-0.9997	-1.0005	-1.0005	1.6266e-04
G04	-30665.5387	-30665.5391	-30665.5387	-30665.5388	-30665.5387	1.3744e-04
G05	5126.4981	5126.4981	5126.4981	5126.4981	5126.4981	2.9308e-06
G06	-6961.8139	-6961.8139	-6961.8139	-6961.8139	-6961.8139	1.1654e-05
G07	24.3062	24.3062	24.3062	24.3062	24.3062	3.4180e-06
G08	-0.0958	-0.0958	0.0000	-0.0607	-0.0580	3.3371e-02
G09	680.6301	680.6301	680.6301	680.6301	680.6301	1.3915e-07
G10	7049.2480	7049.2480	7852.6334	7094.4576	7049.2928	1.7889e+02
G11	0.7500	0.7500	0.7500	0.7500	0.7500	3.0439e-07
G12	-1.0000	-1.0000	-0.6981	-0.9774	-0.9944	6.0681e-02
G13	0.0539	0.0539	1.0000	0.4363	0.4389	2.9314e-01
G14	-47.7611	-47.7611	-47.7611	-47.7611	-47.7611	8.2570e-06
G15	961.7152	961.7152	961.7152	961.7152	961.7152	3.7671e-07
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	1.2884e-11
G17	8876.9807	8853.5411	8927.5980	8862.8537	8857.9908	1.5304e+01
G18	-0.8660	-0.8660	-0.5000	-0.7559	-0.8660	1.3760e-01
G19	32.6556	32.6556	33.8315	32.7255	32.6556	2.3853e-01
G21	193.7869	193.7878	194.0552	193.8408	193.8157	6.4651e-02
G23	-400.0000	-400.0003	-65.5630	-374.3251	-400.0000	8.7152e+01
G24	-5.5080	-5.5080	-4.0537	-4.9995	-5.5080	6.3342e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.0140e-11
SPD	0.0127	0.0127	0.0128	0.0127	0.0127	1.9130e-05
PVD	6059.7143	6059.7143	7476.9170	6233.6224	6092.6044	2.9358e+02
SRD	2994.4710	2994.4696	2994.4711	2994.4706	2994.4706	3.7832e-04

Table C.8: Convergence statistics of RBFI–ESS–SANS–MTRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	369	156	0.42	100	24	1538	156
G02	2495	2473	0.99	100	0	-	2473
G03	1447	1360	0.94	100	96	1507	1360
G04	179	47	0.26	100	100	179	47
G05	185	168	0.91	100	100	185	168
G06	64	54	0.84	100	100	64	54
G07	602	573	0.95	100	100	602	573
G08	112	95	0.85	100	44	254	95
G09	626	597	0.95	100	100	626	597
G10	719	457	0.64	76	12	7886	601
G11	122	90	0.74	100	100	122	90
G12	140	114	0.81	100	40	350	114
G13	375	335	0.89	100	24	1561	335
G14	762	744	0.98	100	100	762	744
G15	166	129	0.78	92	92	196	140
G16	157	114	0.72	100	100	157	114
G17	536	464	0.87	92	88	662	505
G18	689	607	0.88	100	52	1326	607
G19	1701	1643	0.97	100	72	2363	1643
G21	543	511	0.94	88	0	-	580
G23	262	158	0.60	100	92	285	158
G24	65	44	0.68	100	60	108	44
WBD	179	149	0.83	100	100	179	149
SPD	529	518	0.98	100	100	529	518
PVD	408	216	0.53	100	8	5100	216
SRD	301	254	0.84	100	100	301	254
Average	528.23	464.22	0.80	98.00	69.38	1118.46	474.43

C.5 RESB (RBFI–ESS–SANS–BTRS)

Table C.9: Statistical results of the optimal objective function values obtained by RBFI–ESS–SANS–BTRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-13.8281	-14.8594	-15.0000	3.8081e-01
G02	-0.8036	-0.5973	-0.2474	-0.4157	-0.3941	1.0927e-01
G03	-1.0005	-1.0005	-1.0005	-1.0005	-1.0005	7.7526e-06
G04	-30665.5387	-30665.5391	-30665.5387	-30665.5387	-30665.5387	1.2436e-04
G05	5126.4981	5126.4981	5126.4981	5126.4981	5126.4981	8.7666e-07
G06	-6961.8139	-6961.8140	-6961.8139	-6961.8139	-6961.8139	1.4329e-05
G07	24.3062	24.3062	24.3062	24.3062	24.3062	9.7285e-07
G08	-0.0958	-0.0958	-0.0000	-0.0447	-0.0291	3.3551e-02
G09	680.6301	680.6301	680.6301	680.6301	680.6301	1.7502e-07
G10	7049.2480	7049.2480	9329.8183	7250.3919	7049.3146	5.5033e+02
G11	0.7500	0.7500	0.7500	0.7500	0.7500	2.1611e-07
G12	-1.0000	-1.0000	-0.8406	-0.9778	-0.9944	3.6763e-02
G13	0.0539	0.0539	1.0000	0.4201	0.4389	3.0877e-01
G14	-47.7611	-47.7611	-47.7611	-47.7611	-47.7611	1.1884e-06
G15	961.7152	961.7152	961.7152	961.7152	961.7152	1.5995e-07
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	2.6935e-08
G17	8876.9807	8853.5404	8927.6250	8892.6825	8872.3606	3.3841e+01
G18	-0.8660	-0.8660	-0.5000	-0.7949	-0.8660	1.1442e-01
G19	32.6556	32.6556	43.0772	33.4328	32.6557	2.4105e+00
G21	193.7869	193.7857	195.3955	193.8751	193.7875	3.1752e-01
G23	-400.0000	-400.0001	-85.4114	-387.4165	-400.0000	6.1647e+01
G24	-5.5080	-5.5080	-4.0537	-5.1712	-5.5080	5.4757e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.1119e-11
SPD	0.0127	0.0127	0.0159	0.0128	0.0127	6.3846e-04
PVD	6059.7143	6059.7136	51134.6907	8484.2266	6166.7184	9.0159e+03
SRD	2994.4710	2994.4693	2994.4710	2994.4704	2994.4704	4.4867e-04

Table C.10: Convergence statistics of RBFI–ESS–SANS–BTRS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	495	351	0.71	100	88	563	351
G02	2496	2455	0.98	100	0	-	2455
G03	1453	1408	0.97	100	100	1453	1408
G04	257	60	0.23	100	100	257	60
G05	94	61	0.64	100	100	94	61
G06	63	37	0.58	100	100	63	37
G07	693	665	0.96	100	100	693	665
G08	106	66	0.62	100	24	440	66
G09	667	636	0.95	100	100	667	636
G10	774	547	0.71	76	28	3636	720
G11	104	48	0.46	100	100	104	48
G12	127	85	0.67	100	36	353	85
G13	432	378	0.87	96	28	1606	393
G14	853	840	0.98	100	100	853	840
G15	146	96	0.66	92	92	173	104
G16	165	104	0.63	100	100	165	104
G17	595	543	0.91	100	52	1144	543
G18	778	683	0.88	100	68	1143	683
G19	1772	1745	0.98	100	48	3692	1745
G21	355	298	0.84	100	48	739	298
G23	282	173	0.61	100	96	293	173
G24	82	26	0.31	100	64	128	26
WBD	185	138	0.74	100	100	185	138
SPD	371	347	0.93	100	96	387	347
PVD	418	221	0.53	100	12	3485	221
SRD	341	256	0.75	100	100	341	256
Average	542.40	471.76	0.74	98.62	72.31	906.29	479.33

C.6 RESBG (RBFI–ESS–SANS–BTRS–GIS)

Table C.11: Statistical results of the optimal objective function values obtained by RBFI–ESS–SANS–BTRS–GIS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-13.8281	-14.9531	-15.0000	2.2964e-01
G02	-0.8036	-0.5338	-0.2386	-0.3429	-0.2985	8.4126e-02
G03	-1.0005	-1.0005	-1.0005	-1.0005	-1.0005	8.4461e-07
G04	-30665.5387	-30665.5397	-30665.5387	-30665.5388	-30665.5387	2.2197e-04
G05	5126.4981	5126.4981	5126.4981	5126.4981	5126.4981	1.6441e-06
G06	-6961.8139	-6961.8139	-6961.8139	-6961.8139	-6961.8139	4.0774e-06
G07	24.3062	24.3062	24.3062	24.3062	24.3062	5.7721e-09
G08	-0.0958	-0.0958	-0.0000	-0.0533	-0.0376	3.2737e-02
G09	680.6301	680.6301	680.6301	680.6301	680.6301	1.6279e-07
G10	7049.2480	7049.2480	8016.0073	7101.9040	7049.2480	1.9816e+02
G11	0.7500	0.7500	0.7500	0.7500	0.7500	2.6457e-07
G12	-1.0000	-1.0000	-0.8028	-0.9662	-0.9864	5.4802e-02
G13	0.0539	0.0539	1.0000	0.3971	0.4389	3.4712e-01
G14	-47.7611	-47.7611	-47.7611	-47.7611	-47.7611	8.4943e-06
G15	961.7152	961.7152	967.5207	961.9916	961.7152	1.2363e+00
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	1.0532e-08
G17	8876.9807	8853.5416	8927.5978	8884.4230	8868.4329	3.2911e+01
G18	-0.8660	-0.8660	-0.5000	-0.7587	-0.8660	1.2979e-01
G19	32.6556	32.6556	32.7030	32.6586	32.6556	9.9462e-03
G21	193.7869	193.7857	194.1500	193.8308	193.7879	9.4501e-02
G23	-400.0000	-400.0002	-400.0000	-400.0000	-400.0000	4.8238e-05
G24	-5.5080	-5.5080	-4.0537	-5.0436	-5.5080	6.7699e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.1061e-11
SPD	0.0127	0.0127	0.0140	0.0128	0.0127	2.5398e-04
PVD	6059.7143	6059.7144	16066.6570	6577.7310	6146.5597	1.9421e+03
SRD	2994.4710	2994.4695	2994.4711	2994.4705	2994.4706	4.1881e-04

Table C.12: Convergence statistics of RBFI–ESS–SANS–BTRS–GIS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	427	314	0.74	100	96	445	314
G02	2087	2044	0.98	100	0	-	2044
G03	1241	1170	0.94	100	100	1241	1170
G04	262	70	0.27	100	100	262	70
G05	100	60	0.61	100	100	100	60
G06	65	25	0.38	100	100	65	25
G07	620	591	0.95	100	100	620	591
G08	105	60	0.57	100	28	374	60
G09	646	614	0.95	100	100	646	614
G10	573	450	0.79	92	80	778	489
G11	106	38	0.36	100	100	106	38
G12	134	83	0.62	100	40	334	83
G13	422	371	0.88	100	40	1054	371
G14	1070	1048	0.98	100	100	1070	1048
G15	180	104	0.57	84	80	268	123
G16	202	126	0.62	100	100	202	126
G17	575	503	0.87	100	64	899	503
G18	655	545	0.83	100	52	1259	545
G19	1619	1538	0.95	100	80	2023	1538
G21	366	327	0.89	96	40	952	340
G23	294	198	0.68	100	100	294	198
G24	78	31	0.39	100	68	115	31
WBD	179	138	0.77	100	100	179	138
SPD	393	364	0.93	100	80	491	364
PVD	425	198	0.46	100	4	10627	198
SRD	347	242	0.70	100	100	347	242
Average	506.50	432.66	0.72	98.92	75.08	990.07	435.44

C.7 RESBGE (RBFI–ESS–SANS–BTRS–GIS–ETS)

Table C.13: Statistical results of the optimal objective function values obtained by RBFI–ESS–SANS–BTRS–GIS–ETS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5, I_{max} = 1e-8, \Delta_{min,2} = 0.01$)

Prob.	Target	Best	Worst	Mean	Median	Std.
G01	-15.0000	-15.0000	-14.6769	-14.9794	-15.0000	7.2171e-02
G02	-0.8036	-0.6020	-0.2437	-0.3963	-0.3966	1.0349e-01
G03	-1.0005	-1.0005	-1.0005	-1.0005	-1.0005	1.8212e-06
G04	-30665.5387	-30665.5393	-30665.5387	-30665.5387	-30665.5387	1.7696e-04
G05	5126.4981	5126.4981	5126.4988	5126.4981	5126.4981	1.2591e-04
G06	-6961.8139	-6961.8140	-6961.8139	-6961.8139	-6961.8139	2.5056e-05
G07	24.3062	24.3062	24.3062	24.3062	24.3062	9.0971e-07
G08	-0.0958	-0.0958	-0.0000	-0.0486	-0.0291	3.4403e-02
G09	680.6301	680.6301	680.6303	680.6301	680.6301	5.4075e-05
G10	7049.2480	7049.2480	7069.6325	7050.8757	7049.2480	5.2572e+00
G11	0.7500	0.7500	0.7500	0.7500	0.7500	2.5026e-07
G12	-1.0000	-1.0000	-0.8102	-0.9681	-0.9864	4.5660e-02
G13	0.0539	0.0539	1.0000	0.3760	0.4389	2.5044e-01
G14	-47.7611	-47.7611	-47.7602	-47.7610	-47.7611	2.3679e-04
G15	961.7152	961.7152	961.7152	961.7152	961.7152	4.3152e-06
G16	-1.9052	-1.9052	-1.9052	-1.9052	-1.9052	9.2121e-09
G17	8876.9807	8853.5419	8929.6637	8889.4008	8871.1676	3.1802e+01
G18	-0.8660	-0.8660	-0.5000	-0.7915	-0.8660	1.2794e-01
G19	32.6556	32.6556	32.6942	32.6572	32.6556	7.5597e-03
G21	193.7869	193.7859	196.2486	193.9741	193.7906	5.2647e-01
G23	-400.0000	-400.0001	-400.0000	-400.0000	-400.0000	1.9922e-05
G24	-5.5080	-5.5080	-4.0537	-4.9497	-5.5080	6.3702e-01
WBD	1.7249	1.7249	1.7249	1.7249	1.7249	1.1592e-11
SPD	0.0127	0.0127	0.0133	0.0127	0.0127	1.5706e-04
PVD	6059.7143	6059.7143	17375.4773	6660.2567	6121.7221	2.2123e+03
SRD	2994.4710	2994.4701	2994.4711	2994.4709	2994.4710	2.3245e-04

Table C.14: Convergence statistics of RBFI–ESS–SANS–BTRS–GIS–ETS ($\eta_{eco} = 50\%, \delta_g = 1, \delta_f = 10, k_{RES} = 5, I_{max} = 1e-8, \Delta_{min,2} = 0.01$)

Prob.	ANFEs	AREs	TE	FR (%)	SR (%)	ENFEs	EAREs
G01	290	177	0.61	100	92	315	177
G02	2126	2089	0.98	100	0	-	2089
G03	1046	1012	0.97	100	100	1046	1012
G04	148	62	0.42	100	100	148	62
G05	55	43	0.78	100	96	57	43
G06	40	23	0.57	100	100	40	23
G07	485	475	0.98	100	100	485	475
G08	80	54	0.67	100	28	285	54
G09	557	553	0.99	100	92	605	553
G10	532	445	0.84	96	72	769	463
G11	76	51	0.67	100	100	76	51
G12	86	68	0.80	100	16	537	68
G13	360	333	0.92	100	28	1287	333
G14	758	731	0.96	100	84	902	731
G15	101	90	0.88	92	92	120	97
G16	152	121	0.79	100	100	152	121
G17	555	523	0.94	100	60	924	523
G18	503	438	0.87	100	72	698	438
G19	1449	1419	0.98	100	88	1646	1419
G21	327	273	0.84	96	40	851	285
G23	207	138	0.67	100	100	207	138
G24	51	30	0.58	100	52	99	30
WBD	168	154	0.92	100	100	168	154
SPD	404	391	0.97	100	88	459	391
PVD	445	209	0.47	100	8	5568	209
SRD	201	145	0.72	100	100	201	145
Average	430.76	386.27	0.80	99.38	73.38	705.78	387.72

D

Publications

1. C. Liu, Z. Wan, Y. Liu, X. Li, D. Liu, Trust-region Based Adaptive Radial Basis Function Algorithm for Global Optimization of Expensive Constrained Black-box Problems, Structural and Multidisciplinary Optimization. (Under Review)

2. C. Liu, D. Liu and X. Mao, Trust-region Based Successively Refined Radial Basis Function Interpolation Method for Global Optimization of Expensive Large-scale Constrained Black-box Problems, Computer Methods in Applied Mechanics and Engineering (Under review)

3. C. Liu, G. Dong, D. Liu and X. Mao, Adaptive Multi-level Search for Global Optimization: An integrated Swarm Intelligence-metamodelling Technique, Journal of Global Optimization. (Under review)

4. C. Liu, Z. Wan, X. Mao, X. Li, D. Liu, Efficient Strategies for Constrained Black-box Optimization by Intrinsically Linear Approximation (CBOILA), Engineering with Computers.(https://doi.org/10.1007/s00366-020-01160-2).

5. G. Dong, C. Liu, D. Liu and X. Mao, Computationally Efficient Approximations Using Adaptive Weighting Coefficients for Solving Structural Optimization Problems, Mathematical Problems in Engineering. (Accepted, Sep 2020).

6. C. Liu, D. Liu, X. Mao, X. Zhou, Extended Multipoint Approximation Method, In: DEStech Transactions on Engineering and Technology Research, DEStech Publications, Inc., (2017) pp. 219-225.

7. D. Liu, C. Liu, C. Zhang, C. Xu, Z. Du, Z. Wan, Efficient hybrid algorithms to solve mixed discrete-continuous optimization problems: A comparative study, Eng. Comput. (Swansea, Wales). 35 (2018) pp. 979-1002.